

AD-A073 133

COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON DC
WORLDWIDE MILITARY COMMAND AND CONTROL SYSTEM (WWMCCS). H-6000 --ETC(U)
SEP 78 B M WALLACK, G H GERO
CCTC-TM-180-78-VOL-3

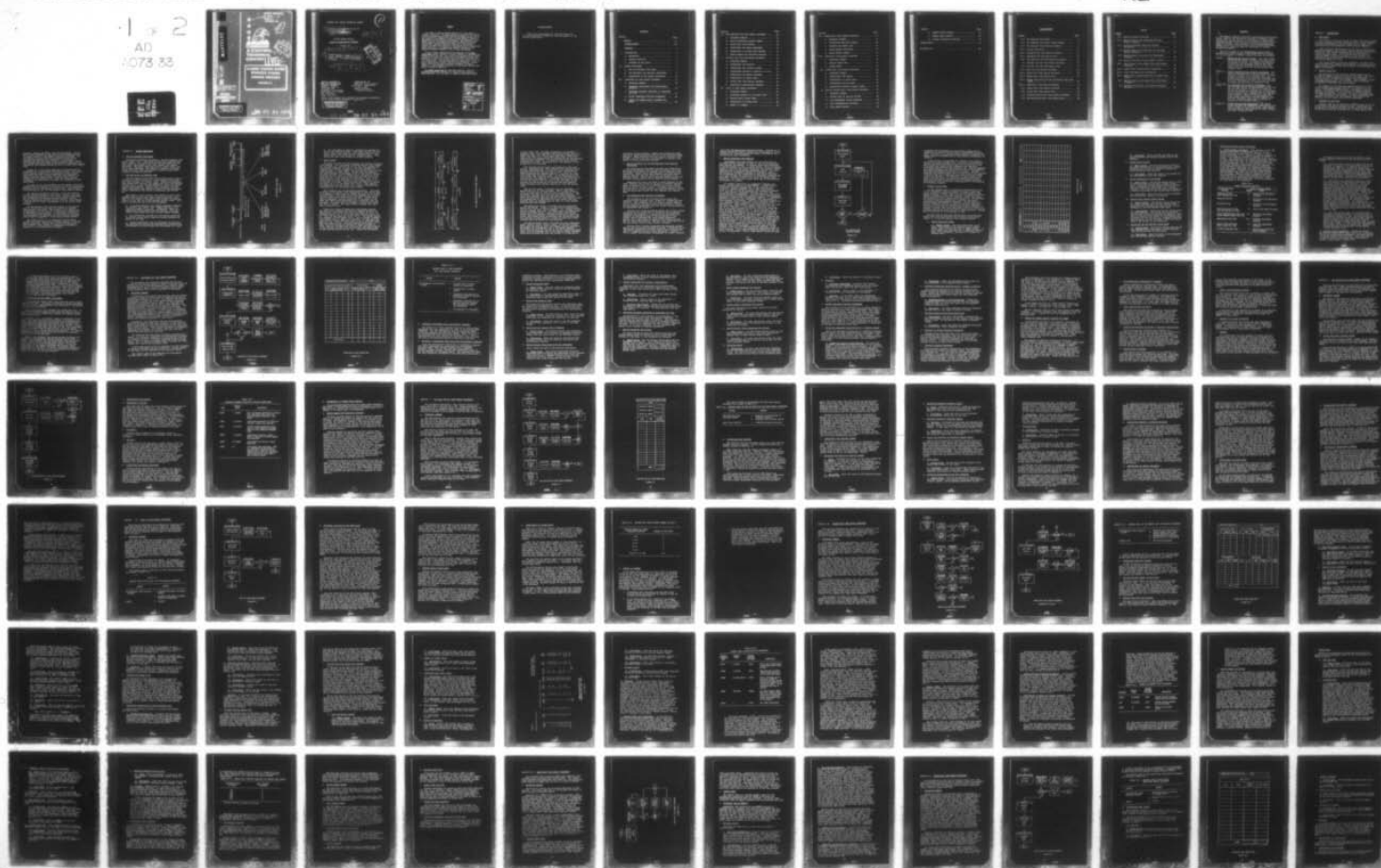
F/G 9/2

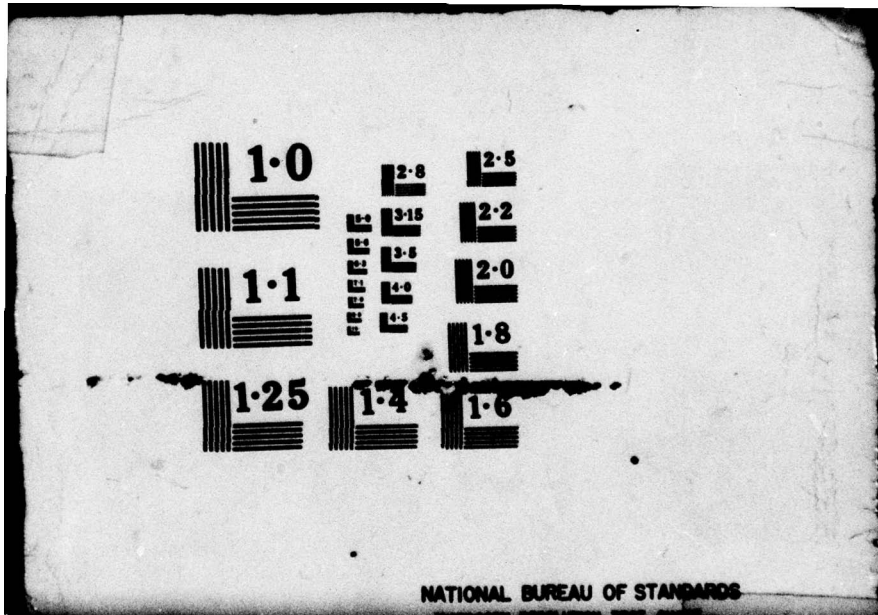
UNCLASSIFIED

NL

1 of 2

AD
A073 133





ADA 073133

DDC FILE COPY

COPIES OF THIS DOCUMENT MAY
OBTAINED FROM THE DEFENSE
DOCUMENTATION CENTER,
MERCH STATION, ALEXANDRIA
EGYPT

DEFENSE
COMMUNICATIONS
AGENCY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

★ AD 59938

TECHNICAL MEMORANDUM
TM 180-78
VOLUME III
1 SEPTEMBER 1978



COMMAND
& CONTROL
TECHNICAL
CENTER

LEVEL III

D'D'C
REFRAME
AUG 24 1978
C

H-6000 TUNING GUIDE
WWMCCS SYSTEM
TUNING PROCESS

VOLUME III

22 07 24 095

COMMAND AND CONTROL TECHNICAL CENTER

9 Technical Memorandum TM 180-78

11 1 September 1978

12 110p.

H-6000 TUNING GUIDE

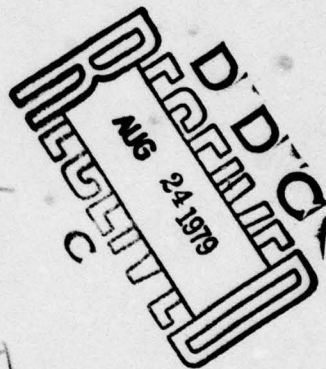
~~WWMCCS SYSTEM TUNING PROCESS~~

Volume III.

14 CCTC-TM-180-78-VOL-3

6 Worldwide Military Command and Control System (WWMCCS). H-6000 Tuning Guide. Volume III. TSS Response Time Analysis Procedures.

10 Barry M. /Wallack
George H. /Gero



PROJECT PERSONNEL

Barry M. Wallack
BARRY M. WALLACK
GEORGE H. GERO JR.
CCTC/CPE/C702
Rm BE685, The Pentagon
Washington, D.C. 20301
AV - 692-2725
COMM - 697-2725

APPROVED BY:

Samuel D. Purnace
For TIPTON P. MOTT-SMITH
Colonel, USAF
Deputy Director,
Computer Services

Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314.

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

226<

4/09 658

79 07 24 095

PREFACE

This report is based on detailed analysis of a large amount of technical information. The results address procedures for the analysis of batch turnaround time and GCOS Time Sharing System response time in World Wide Military Command and Control Systems (WWMCCS). Because of the complexity of the analysis procedures and their dependence on the WWMCCS workloads and operational environments, generalizing the procedures beyond the environment described or extracting conclusions without their respective qualifying conditions is not possible. Questions related to this report or to the possibility of extending the stated conclusions or recommendations should be addressed to the Computer Performance Evaluation Office, (C702), the Pentagon.

To gain a general understanding of the approach of the H-6000 Tuning Guide, Volume I Section II, Volume II Section II, and Volume III Section II should be read. One or more of the hypothesis tests (search procedures) in Volume II Sections IV-XII and Volume III Sections III-X should also be read. Not all these tests have to be read at the start of a tuning effort. Each should be read as it needs to be applied. To start a tuning effort, Volume I should be read and applied. The procedure for analysis of batch turnaround time begins in Volume II Section III. The procedure for analysis of Time Sharing response time begins in Volume III Section II.

The H-6000 Tuning Guide has never been tested by a novice in performance evaluation, although field tests have been conducted by FEDSIM personnel. For this reason, it remains a preliminary version.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By PER LETTER	
Distribution/	
Availability Codes	
Dist	Avail and/or special
A	

ACKNOWLEDGMENT

This is to acknowledge the time and effort of
Dr. John Peterson of FEDSIM in the development of the
H-6000 Tuning Guide.

CONTENTS

Section	Page
PREFACE	ii
ACKNOWLEDGMENT	iii
ABSTRACT.	x
I. INTRODUCTION	1
A. BACKGROUND	1
B. PROJECT OBJECTIVE	1
C. CONTENTS OF THE GUIDE	1
II. SEARCH PROCEDURES	3
A. THE TSS RESPONSE TIME MODEL	3
B. TSS RESPONSE TIME ANALYSIS PROCEDURES	8
C. CONSTRAINTS TO THE SEARCH PROCEDURES	15
III. SUBSYSTEM CPU TIME SEARCH PROCEDURE	16
A. PROCEDURE SUMMARY	16
B. DETERMINE SUBSYSTEMS WITH UNACCEPTABLE RESPONSE	19
C. DETERMINE DOMINANT SUBSTATES OF SUBSYSTEM CPU TIME	21
D. ADJUST SUBSYSTEM DISPATCH PARAMETERS	23
E. REDUCE CPU DEMAND AND/OR INCREASE CPU POWER	25

Section	Page
IV. TSS EXECUTIVE CPU TIME SEARCH PROCEDURE	28
A. PROCEDURE SUMMARY	28
B. ADJUST SUBSYSTEM DISPATCH LENGTH	28
C. ADJUST WAIT TIME PERIODS	30
D. INVESTIGATE SSA MODULE RESIDENCE	30
E. INVESTIGATE I/O QUEUE SPACE DENIALS	32
F. REDUCE DEMAND FOR EXECUTIVE SERVICES	32
V. TSS WAIT FOR CPU TIME SEARCH PROCEDURE	33
A. PROCEDURE SUMMARY	33
B. INVESTIGATE TSS PRIORITY	36
C. INVESTIGATE TSS DISPATCH LENGTH	37
D. INVESTIGATE PRIORITY B DISPATCH BLOCKING .	40
E. INVESTIGATE SSA MODULE RESIDENCE	40
F. INVESTIGATE I/O QUEUE SPACE	41
G. INVESTIGATE LINE SERVICE INTERVAL	42
H. INVESTIGATE INTERMITTENT PROBLEMS	42
VI. DISK I/O TIME SEARCH PROCEDURE	44
A. PROCEDURE SUMMARY	44
B. DETERMINE LOCATION OF TSS WORK FILES	46
C. EXECUTE BATCH VOLUME TESTS	47
D. INVESTIGATE I/O QUEUE SPACE	48
E. REDUCE I/O DEMAND	49

Section	Page
VII. MEMORY WAIT TIME SEARCH PROCEDURE	51
A. PROCEDURE SUMMARY	51
B. INVESTIGATE SWAPPING PROBLEM	54
C. INCREASE TSS MEMORY SIZE	58
D. ADJUST MEMORY PRIORITIES	69
E. REDUCE MEMORY DEMAND	74
VIII. GWAKE WAIT TIME SEARCH PROCEDURE	76
A. PROCEDURE SUMMARY	76
B. ANALYZE GWAKE TIME	76
C. TUNING STEPS	78
IX. OUTPUT WAIT TIME SEARCH PROCEDURE	80
A. PROCEDURE SUMMARY	80
B. INVESTIGATE USER ERRORS	82
C. INVESTIGATE I/O QUEUE SPACE	84
D. INVESTIGATE OUTPUT VOLUME	84
E. INVESTIGATE POSSIBLE DATANET DELAYS	85
X. NON-TSS SERVICE WAIT TIME SEARCH PROCEDURE ...	87
A. PROCEDURE SUMMARY	87
B. ISOLATE TYPE OF NON-TSS PROCESS	88
C. FILE MANAGEMENT SYSTEM PROCESSES	92
D. GCOS INTERROGATION PROCESSES	93
E. LINE LENGTH PROCESS	94

Section	Page
F. SPECIAL BATCH PROCESS	94
G. NORMAL BATCH PROCESS	95
H. CONSOLE INTERACTION PROCESS	95
DISTRIBUTION	97
DD FORM 1473	98

ILLUSTRATIONS

Figure	Page
II-1 TSS Response Time Model	4
II-2 TSS Response Time Model States Example	6
II-3 TSS Response Time Analysis Summary	9
II-4 Initial Data Form	11
III-1 Subsystem CPU Time Search Procedure	17
III-2 Subsystem CPU Time Search Form	18
IV-1 TSS Executive CPU Time Search Procedure	29
V-1 TSS Wait for CPU Time Search Procedure	34
V-2 TSS Wait for CPU Time Search Form	35
VI-1 Disk I/O Time Search Procedure	45
VII-1 Memory Wait Time Search Procedure	52
VII-2 Memory Wait Time Search Form	55
VII-3 Sample Calculation of the Percentage Less Than .TASID	62
VIII-1 GWAKE Wait Time Search Procedure	77
IX-1 Output Wait Time Search Procedure	81
IX-2 Output Wait Time Search Form	83
X-1 Non-TSS Service Wait Time Search Procedure ...	89
X-2 Non-TSS Service Wait Time Search Form	90

TABLES

Number		Page
II-1	Search Procedure Sections	13
III-1	Reports Used in the Subsystem CPU Time Search Procedure	19
IV-1	Intervals Between Executive Routine Executions	31
V-1	Reports Used in the TSS Wait for CPU Time Search Procedure	36
VI-1	Reports Used in the Disk I/O Time Search Procedure	44
VI-2	Maximum TSS Users versus Number of SSA's	49
VII-1	Reports Used in the Memory Wait Time Search Procedure	54
VII-2	Urgent User Classification Parameters	64
VII-3	Other TSS Memory Size Parameters	68
VII-4	Memory Wait Periods Required for Urgent User Status	73
IX-1	Reports Used in the Output Wait Time Search Procedure	82
X-1	Non-TSS Service Wait Time Search Procedure Sections	92

ABSTRACT

The Federal Computer Performance Evaluation and Simulation Center (FEDSIM) has developed a document for WWMCCS installations that can be used by site personnel to analyze the performance characteristics of their Honeywell 6000 (H-6000) computer systems. This document, called an H-6000 Tuning Guide, incorporates detailed analysis procedures that guide the analyst in applying specific techniques to improve system performance.

The four volumes of the H-6000 Tuning Guide present a precisely structured system of procedures for the analysis of the performance of WWMCCS computer services and systems:

- Volume I WWMCCS System Tuning Process. The first volume describes the overall structure and application of the Tuning Guide. It explains the approach, procedures, and processes taken by the Tuning Guide to provide analyses of batch job turnaround time and GCOS Time Sharing System (TSS) response time.
- Volume II Batch Turnaround Time Analysis Procedures. The second volume presents a set of procedures for analysis of batch job turnaround time. It first presents a model of the processes and queue points associated with batch job turnaround time and then describes nine tests that use the model to direct the analysis of turnaround time.
- Volume III TSS Response Time Analysis Procedures. The third volume serves the same general purpose and has the same general structure as Volume II. Volume III presents a complete set of procedures for investigating the response time of GCOS Time Sharing System (TSS) interactions. The volume first presents a model of the processes and queue points associated with TSS response time and then describes eight tests to direct an analysis of TSS response time.
- Volume IV H-6000 Tuning Guide Appendices. The fourth volume provides the appendices referenced by the other volumes of the Tuning Guide. The volume contains detailed descriptions of report formats and other referenced data.

SECTION I. INTRODUCTION

A. BACKGROUND

The Office of the Joint Chiefs of Staff (JCS) has directed that the Command and Control Technical Center (CCTC) develop a computer performance analysis capability to support the World Wide Military Command and Control System (WWMCCS).

CCTC, acting at the direction of the JCS, specified that WWMCCS ADP managers should apply various computer performance evaluation (CPE) tools and techniques to the systems now running at their sites. CCTC also defined the need to instruct WWMCCS technical personnel in the selection and application of the CPE tools and techniques appropriate to individual WWMCCS ADP sites.

CCTC asked FEDSIM to plan and implement a document that could be employed by WWMCCS ADP personnel to diagnose problems and to propose changes that would improve the performance of WWMCCS ADP systems.

B. PROJECT OBJECTIVE

The objective of the resulting FEDSIM project was to provide all WWMCCS installations with a document that could be used by staff personnel to analyze the performance characteristics of their ADP systems. This document, called an H-6000 Tuning Guide, was to contain sets of analysis procedures to improve system performance.

The product of the completed FEDSIM project is a four-volume H-6000 Tuning Guide (referred to hereafter as the Guide). The Guide volumes present a precisely structured system of procedures for the analysis of WWMCCS computer services and systems. The titles of the four volumes are: (1) WWMCCS System Tuning Process, (2) Batch Turnaround Time Analysis Procedures, (3) TSS Response Time Analysis Procedures, and (4) H-6000 Tuning Guide Appendices.

C. CONTENTS OF THE GUIDE

Computer jobs may be submitted by WWMCCS users to run as either batch jobs or as interactive jobs. Batch jobs, as processed by the WWMCCS systems, may be submitted by users at the site or may be initiated via a process called "job

spawning" through the WWMCCS Time Sharing System. Interactive jobs addressed by the Guide are the subsystems that run under control of the WWMCCS Time Sharing System. The performance of both batch and interactive jobs can be measured and analyzed with reference to the amount of elapsed time that the system takes to process them. Batch job elapsed processing time is called batch turnaround time. Interactive job elapsed processing time is called TSS response time.

Volume III of the Guide presents an integrated set of procedures for investigating TSS response time. The volume first presents a model of the processes and wait states associated with TSS response times and then describes an overall plan and eight search procedures that direct an analysis of TSS response time. Both the model and the procedures are specifically designed for the WWMCCS system environment.

In order to use the procedures in this Guide it will be necessary for the analyst to be familiar with the Generalized Monitor Facility Time Sharing Tuning Guide - Data Collector, the TSTAT Monitor and the TSS Response Time Analysis System.

The Generalized Monitor Facility is the tool used to collect the data needed by this guide. Without that data, the procedures described in this guide cannot be executed. All information for running the data collector can be found in the Generalized Monitoring Facility - Users Manual - CSM UM 246-78 1 April 78 & appendix J of the Tuning Guide Document.

The TSS Response Time Analysis System is the program used to reduce the data tapes and produce all reports referenced in this guide. The format of all reports and an explanation of the reports is contained in appendix H of the Tuning Guide Document. A description of the run-time procedures for operating the data reduction program can be found in Appendix J of the Tuning Guide Document.

The TSTAT Monitor is an independent monitor which gathers statistics from within internal tables of the Time Sharing Subsystem, formats these statistics and prints out a report. This program is completely described in Appendix K of the Tuning Guide Document.

SECTION II. SEARCH PROCEDURES

A. THE TSS RESPONSE TIME MODEL

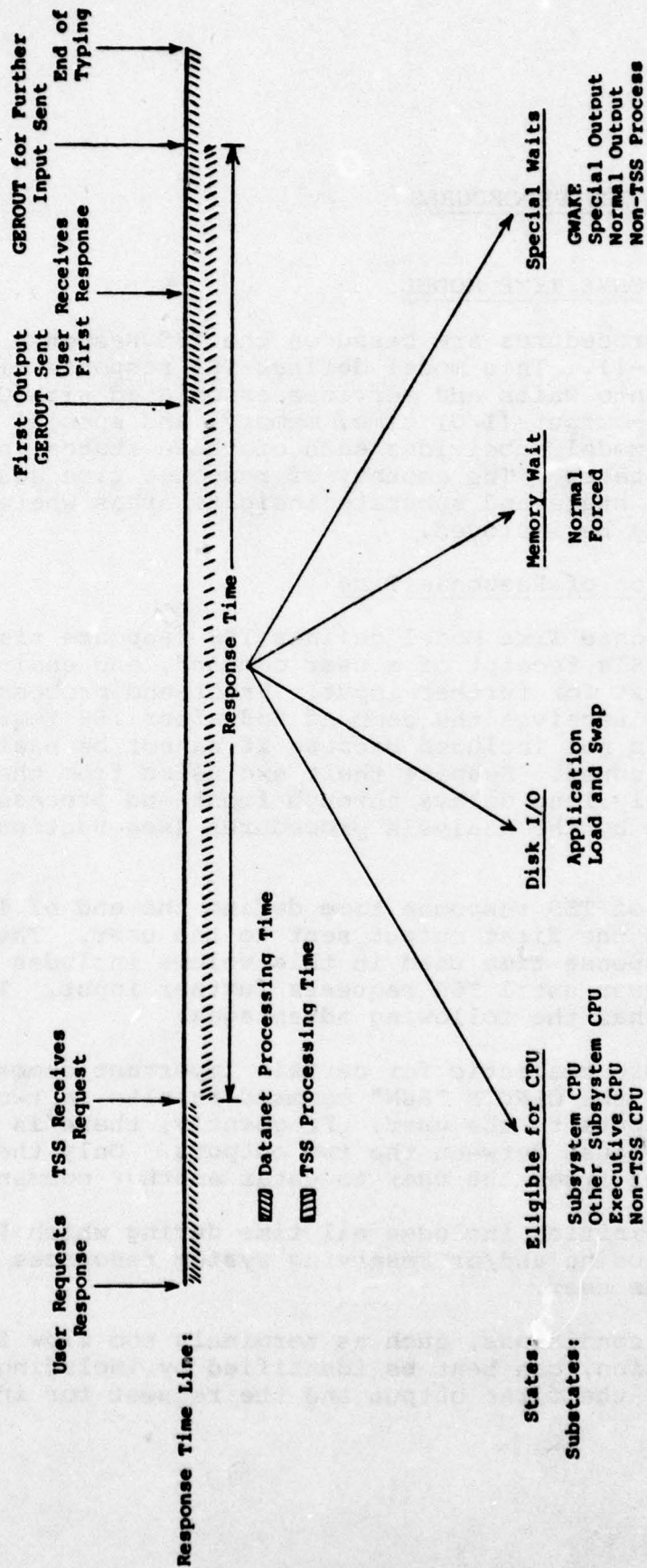
The Search Procedures are based on the TSS Response Time Model (Figure II-1). This model defines TSS response time and divides it into waits and services associated with CPU time, disk input-output (I/O) time, memory, and special processes. The model subdivides each of these states into two or more substates. The amounts of response time associated with each state and substate indicate areas where response time may be improved.

1. The Definition of Response Time

The TSS Response Time Model defines TSS response time as starting with TSS's receipt of a user command, and ending with TSS's request for further input. Front-end processing time (before TSS receives the command and after TSS requests further input) is not included because it cannot be easily and reliably measured. Despite their exclusion from the model, excessively long delays through front-end processing can be indicated by the analysis procedures (see Section IX).

Many models of TSS response time define the end of TSS response time as the first output sent to the user. The model of TSS response time used in this volume includes all outputs to the user until TSS requests further input. This extra inclusion has the following advantages:

- a. It is more realistic for certain important commands. For example, the CARDIN "RUN" command results in two separate outputs to the user. Frequently, there is a significant pause between the two outputs. Only the second output frees the user to enter another command.
- b. This definition includes all time during which TSS is actively using and/or reserving system resources on behalf of the user.
- c. Certain conditions, such as terminals too slow for the application, can best be identified by including the time between the first output and the request for input.



TSS RESPONSE TIME MODEL

FIGURE II-1

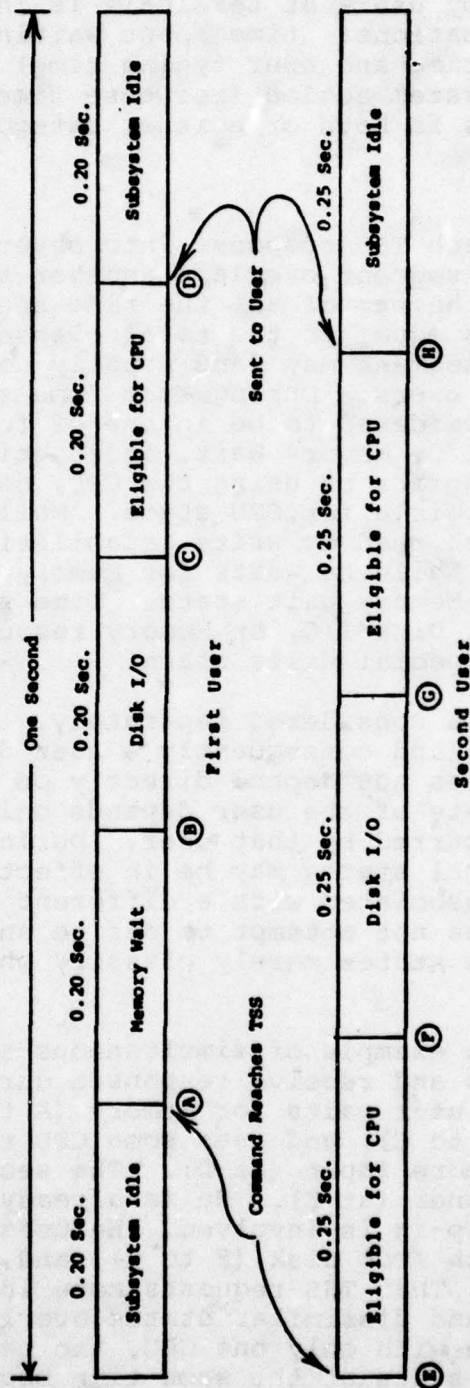
d. All time spent by users at terminals is included in one of two classifications: time spent waiting for user action (user think time and user typing time) and time spent waiting for system action (response time). There is no time that fits in both or neither category.

2. Model States

The model divides each TSS response into several different time segments. No time segment overlaps another time segment for the same user, and the sum of all the time segments for a particular response is equal to the total elapsed TSS response time. A time segment may (and usually does) overlap time segments for other users. During each time segment, the user involved is considered to be in one of four states: Eligible for CPU, Disk I/O, Memory Wait, and Special Waits. While the user is waiting for or using the CPU, he is considered to be in the Eligible for CPU state. While he is waiting for or using disk read or write capabilities, he is in the Disk I/O state. While he waits for memory to become available, he is in the Memory Wait state. Time segments not associated with CPU, Disk I/O, or Memory resources are considered part of the Special Waits state.

Each response time is considered separately. The state to which a time segment (and consequently a user during that time segment) belongs does not depend directly on the states of other users. The state of the user depends only on the type of delays being incurred by that user. During simultaneous responses, several states may be in effect at the same time, each state associated with a different user and response. The model does not attempt to define any states for TSS as a whole. The states merely classify what each user is trying to do.

Figure II-2 gives an example of simultaneous states. Two users enter commands and receive responses during the same second. The first user waits for memory (A to B), is swapped in from disk (B to C), and uses some CPU time (C to D). Then TSS requests more input (at D). The second enters his command a little sooner (at E). He is already in memory, so no memory wait or swap-in is involved. He uses some CPU time (E to F), reads data from disk (F to G), and uses some more CPU time (G to H). Then TSS requests more input (at H). Note that both similar and dissimilar states overlap in time. Even on a machine with only one CPU, two users may be in the Eligible for CPU state at the same time because one or both may be waiting for the CPU. During the one second



TSS RESPONSE TIME MODEL STATES EXAMPLE

FIGURE II-2

of wall clock time, the model recognized 1.35 seconds of response time: 0.70 seconds of Eligible for CPU time, 0.45 seconds of Disk I/O time, and 0.20 seconds of Memory Wait time. The time (i.e., user think time) that is not considered part of response time is reported by the TSS Response Time Analysis System as if it were a fifth state called Subsystem Idle time. In Figure II-2, 0.65 seconds of Subsystem Idle time occurred.

The Eligible for CPU state is divided into four substates: Subsystem CPU, Other Subsystem CPU, Executive CPU, and Non-TSS Process CPU. A user is in the Subsystem CPU substate when he is actually using a CPU. He is in the Other Subsystem CPU substate when he is waiting for a CPU and another user is using a CPU. If the TSS executive is using a CPU, all users in the Eligible for CPU state are in the Executive CPU substate. If the TSS executive is not using a CPU and no user is using a CPU, all users in the Eligible for CPU state are in the Non-TSS CPU substate. These substates help isolate reasons for long CPU service times.

The Disk I/O state is divided into two substates: (1) Application and (2) Load and Swap. When a disk read or write is being used to swap the user's subsystem in or out of memory, the user is in the Load and Swap substate. All other disk accesses fall into the Application substate. These two substates indicate whether long disk service times are related only to TSS executive files (swap files and program files) or whether they are part of a more general disk service time problem.

The Memory Wait state is divided into two substates: Normal and Forced. A user is in the Forced substate whenever he has just been "force swapped", i.e., swapped because he was in memory too long. All other waits for memory are treated as Normal waits. Separating Normal from Forced memory waits is important, because Forced waits tend to be longer than Normal waits and because long Forced waits may be tolerated at some sites as a means of penalizing those commands that use memory for long periods.

The Special Waits state is divided into four substates: GWAKE, Special Output, Normal Output, and Non-TSS Process. Those users waiting by their own request (having executed a DRL GWAKE) are considered to be in the GWAKE substate. Those users waiting for the end of certain executive outputs (such as error messages) are in the Special Output substate. Those users waiting for their own output to finish are in

the Normal Output substate. Those users waiting for some task to be performed outside TSS are in the Non-TSS Process substate. These substates are not closely related to one another, and their division is necessary to separate indicators of vastly different problems.

3. Use of the Model by the TSS Response Time Analysis Procedures

The TSS Response Time Model provides the framework on which the TSS Response Time Analysis Procedures are based. The model's definition of response time is assumed by the procedures. Model states and substates with high percentages of response time are chosen for investigation and possible tuning. These states represent general areas that are then investigated by one or more of the procedures.

Comparing the total time spent in each substate to the total response time allows an analyst to estimate improvements in response that can be achieved by tuning various parts of TSS or the host system. The model cannot predict the results of a tuning effort, but applied properly, it can provide approximate bounds for TSS response time improvement.

B. TSS RESPONSE TIME ANALYSIS PROCEDURES

The procedures in the following section begin the analysis of TSS response time. Figure II-3 summarizes the analysis. The procedures in this section (except Section II.B.1) should not be applied until the problem is defined, the environment understood, and the current tuning objective is formulated (see Volume I).

Section II.B.1 must be applied before the current tuning objective can be formulated; it discusses how to define acceptable TSS response. A definition of acceptable TSS response is assumed by many of the procedures in this Volume, and is necessary for their proper execution. Section II.B.2 describes the method of gathering initial data to start the analysis. Section II.B.3 describes how to use the initial data to start further analysis.

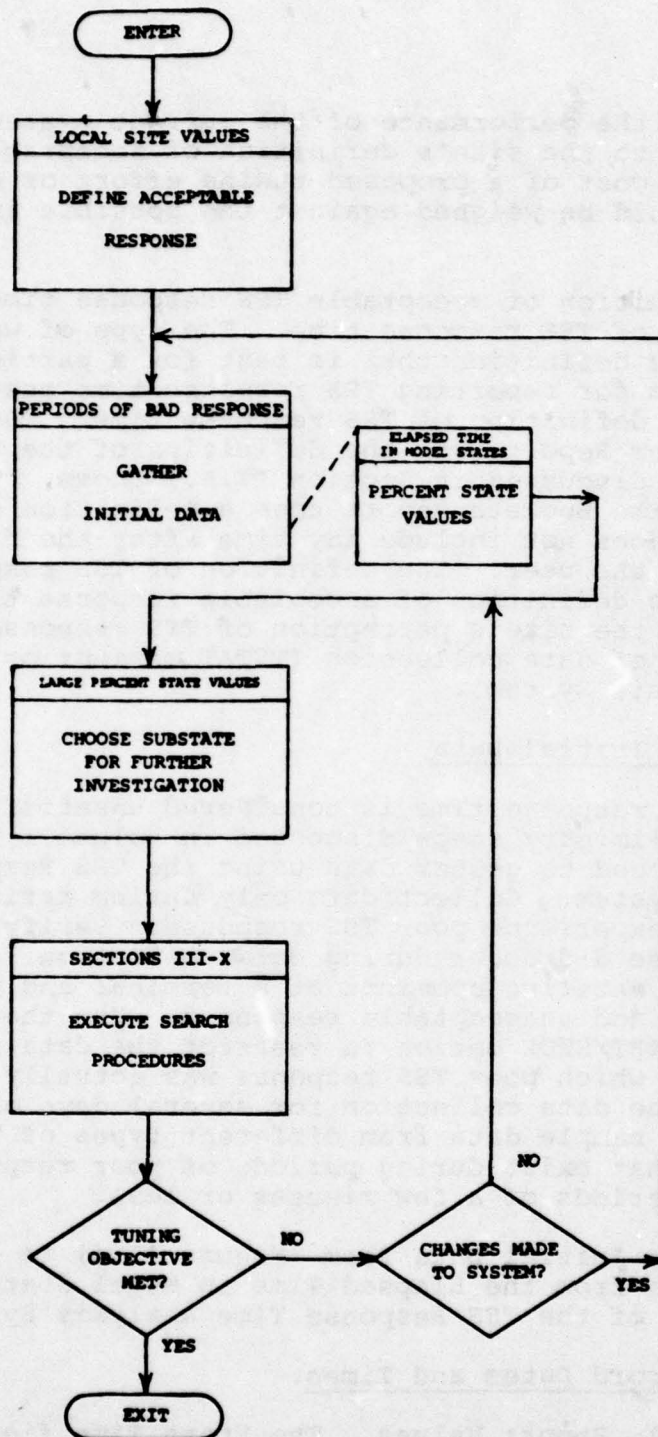
Section II.B.1 (Define Acceptable TSS Response) covers part of the activities discussed in Volume I, Sections III.C (Understand Installation Service Objectives and Priorities) and III.D (Specify Current Tuning Objective). Section II.B.2 (Gather Initial Data) corresponds to Volume I Section

IV.B (Run the Appropriate Analysis System). Section II.B.3 (Choose and Execute Search Procedures) corresponds to Volume I Section IV.C (Evaluate Analyzer Output) and Section IV.D (Follow Guide Test Procedures).

1. Define Acceptable TSS Response

Each WWMCCS site must develop its own definition of satisfactory TSS response in order to use the TSS Response Time Analysis Procedures. An analyst needs this definition to decide whether to start a tuning effort, and he frequently needs it to define the tuning objective (for example, reduce response time to site-defined acceptable level). The site definition of acceptable response is frequently used in the procedures to make investigation and tuning decisions. If no well-thought out definition of acceptable response has been made, no clear criteria will exist for making these decisions. The resulting tuning effort may be disorganized and blunted in purpose.

No standard definition of acceptable response time exists, either inside or outside WWMCCS. A definition of acceptable response depends both on the workload (e.g., trivial or lengthy commands) and on user requirements. Since the goal of the TSS Response Time Analysis Procedures is to help produce acceptable response, the site must determine what is acceptable. An unrealistically optimistic definition would necessitate expensive and/or time-consuming measures to achieve that "acceptable response." A pessimistic definition would keep site management from recognizing a true response time problem. Defining acceptable response as "all responses under X seconds" is also unlikely to be satisfactory. Some commands involve much more work (I/O and CPU time) than others. One possible definition is that 90% of the response times are below a specified value and that the average response for all (or the longest 10%) is below another specified value. A better definition might involve different response limits for different commands -- X seconds for responses to the RUN command, Y seconds for certain kinds of data base searches, etc. TSS response can be checked with the TSS Response Time Buckets Report (known to the TSS Executive as the TSS Response Time Histogram) of the TSTAT monitor. It may also be checked with the TOTAL line on the TSS Response by Subsystem Report of the TSS Response Time Analysis System. If a site uses different response limits for different workloads/commands, it may be able to use the full TSS Response by Subsystem report to check response. Each separate workload/command may use its own set of subsystems. One of these reports should be used



TSS RESPONSE TIME
ANALYSIS SUMMARY

FIGURE II-3

to measure the performance of the current system and to compare it to the site's definition of acceptable response time. The cost of a proposed tuning effort or change to the system should be weighed against the possible improvement in response.

A definition of acceptable TSS response time assumes a definition of TSS response time. The type of workload may dictate the definition that is best for a particular site. Each system for reporting TSS response time assumes a particular definition of TSS response time. The TSS Response by Subsystem Report uses the definition of the TSS Response Time Model discussed in Section II.A.1 above. The TSS Response Time Buckets Report uses a definition of TSS response time that does not include any time after the first output is sent to the user. The definition of TSS response time used in the definition of acceptable response time must agree with the site's perception of TSS response and with the method of data collection (TSTAT monitor or TSS Response Time Analysis System).

2. Gather Initial Data

If TSS response time is considered unsatisfactory, and if the preliminary steps discussed in Volume I have been taken, proceed to gather data using the TSS Response Time Analysis System. Collect data only during periods which are likely to experience poor TSS response. Verify that poor TSS response did occur during data collection. If necessary, do this by entering commands at a terminal and noting acceptable and unacceptable responses. Use the data reduction START/STOP option to restrict the data reported to periods in which poor TSS response was actually observed. Continue the data collection for several days or longer, if needed, to sample data from different types of TSS and batch workload that exist during periods of poor response. Discard data for periods of a few minutes or less.

Use the Initial Data form (Figure II-4) to record pertinent data from the Elapsed Time in Model States report (Report 1) of the TSS Response Time Analysis System.

a. Record Dates and Times.

(1) Report Values. The Start Time field in the upper right corner of the Elapsed Time in Model States report records the date in MMDDYY format. It also records the starting time in HHMM.ffff format, where ffff represents a fraction of a minute in decimal form.

(2) Form Entry. Enter the date and time in the proper rows for each separate period of poor TSS response.

b. Record Period Length

(1) Report Value. The TSS Activity For field in the upper right corner of the Elapsed Time in Model States report records the number of seconds for which TSS traces were received.

(2) Form Entry. Enter the value in the Length row for each period of poor TSS response.

c. Record Subsystem CPU Percentage

(1) Report Value. The Percent States value in the SS CPU row of the Elapsed Time in Model States report represents the percentage of total terminal time user subsystems were using a CPU. Total terminal time is the sum of all time users spent at a terminal connected to TSS during data collection.

(2) Form Entry. Enter the value in the Subsystem CPU row for each reported time period.

d. Record Other Percent States Values

(1) Report Values. The Percent States values for the other substates below SS CPU represent the percentages of terminal time spent in the various substates of the TSS Response Time Model.

(2) Form Entries. Record the Percent States value for each substate in the proper row of the form. Record these values for each period reported. Note that the Percent State values for the model states (Eligible for CPU, SS Disk I/O, SS Memory Wait, and Special Waits) are not recorded. These are simply sums of the values for the substates.

e. Record the SS Idle Percent State Value

(1) Report Value. The Percent States value for SS Idle represents the percentage of total terminal time TSS was waiting for user input.

(2) Form Entry. Enter the value in the Subsystem Idle row for each period reported.

3. Choose and Execute Search Procedures

a. Choose Search Procedures. Each substate of the TSS Response Time Model is associated with one of the Search Procedures in Sections III through X of this volume (see Table II-1). Choosing a search procedure to follow is the same as choosing a substate to investigate. The substates to be investigated would usually be those associated with the largest percentages recorded on the Initial Data form. If one substate shows the highest percentage value for each time period, follow the search procedure associated with that substate. If a few substates always have the highest percentages, follow the search procedure (s) associated with each of these substates. Following more than one procedure simultaneously may reduce the time and effort needed for the study. Special attention should be given to investigations or tuning steps proposed by more than one of the search procedures being followed. These investigation or tuning steps may reduce simultaneously the time spent in two substates.

TABLE III-1.

SEARCH PROCEDURE SECTIONS

RESPONSE TIME MODEL SUBSTATE	SEARCH PROCEDURE SECTION	
	NUMBER	TITLE
Subsystem CPU Time Other Subsystem CPU Time	III	Subsystem CPU Time Search Procedure
Executive CPU Time	IV	TSS Executive CPU Time Search Procedure
Non-TSS CPU Time	V	TSS Wait for CPU Time Search Procedure
Application Disk I/O Time Load and Swap Disk I/O Time	VI	Disk I/O Time Search Procedure
Normal Subsystem Memory Wait Time Forced Subsystem Memory Wait Time	VII	Memory Wait Time Search Procedure
GWAKE Special Wait Time	VIII	GWAKE Wait Time Search Procedure
Special Output Wait Time Normal Output Wait Time	IX	Output Wait Time Search Procedure
Non-TSS Process Wait Time	X	Non-TSS Service Wait Time Search Procedure

An analyst may want to modify his choice of procedures somewhat according to the following considerations:

(1) Relatively small amounts of GWAKE time may be important if they occur in subsystems which do not normally use GWAKE's. In addition, some sites occasionally show large amounts of system resources consumed by status subsystems such as DJST. The amount of resources consumed by these subsystems can frequently be cut with little degradation of service to the users. Execute the GWAKE Wait Time Search Procedure to investigate these problems.

(2) Occasional long time periods spent in a substate may warrant investigation even though the percentage value for that substate is not high. These occasional long time periods may cause response to be erratic and thus unacceptable. Identify occasional long time periods using the Standard Deviation and Maximum Value columns of the Elapsed Time in Model States report. Verify that the occasional long time periods occur in all periods of poor response. Execute the search procedure associated with that substate, keeping in mind that occasional long time periods are the problem being investigated. For example, an investigation of occasional long disk service times would emphasize disk device errors and I/O queue space exhaustion, both of which could cause occasional long service times without affecting all disk service times.

(3) Long time periods spent in the Memory Wait substates (Forced and Normal) may be the result of bottlenecks elsewhere in TSS. Alleviating bottlenecks in CPU and disk service and in some types of Non-TSS Process will cause subsystems to be in memory for shorter periods and so will tend to lower memory wait times. If other substates show percentages almost as high as the memory wait substates, an analyst may wish to investigate the non-memory related substates first.

b. Execute Search Procedures. Execute the search procedure(s) chosen above either singly or in parallel. Follow any tuning steps recommended. Evaluate whether the Tuning Objective has been met. This evaluation is a separate step in the tuning effort described in Volume I (see Volume I, Figure II-1).

If the tuning effort is to be continued, obtain initial data and choose search procedures again. Only if no tuning steps were taken and the workload has not changed can the original initial data be used again. Choose search procedures again, according to the percentages recorded on the form. If the procedures chosen were previously executed, redundant steps may be skipped and different decision paths taken. Any tuning steps made when the procedures were previously executed may have seemingly unrelated influences on which decision path to take. Care should be taken that changes in the data are not overlooked.

C. CONSTRAINTS TO THE SEARCH PROCEDURES

The procedures cannot cover every possible cause of poor TSS response time. Including procedures to handle improbable cases is costly, time-consuming, and confusing to the analyst following the procedures.

The procedures in this document are guidelines only. If acceptable response cannot be achieved with them or if the data fit none of the conditions treated, the analyst should seek outside help.

Brackets (i.e., []) around a value or expression in the text signal that the value or expression is suggested only. These suggested values are used in the search procedures to avoid vagueness about decision making; but, because these values remain suggestions, they will not be correct in every case. If a situation does not seem to "fit" the assumptions, or if a decision value is nearly equal to the bracketed expression, one of two actions may be expedient: (1) explore both logical paths leading from the decision in question, and/or (2) seek outside help.

Not all the parameter and system changes discussed are expected to significantly affect response time at most sites. These changes are included because they may significantly affect certain systems under certain workloads with certain sets of site parameter values. At any one site, many of the changes discussed may produce little improvement to response. Therefore, the analyst should not labor too long over one part of the procedure and neglect parts that might be much more helpful.

SECTION III. SUBSYSTEM CPU TIME SEARCH PROCEDURE

The procedures for analyzing subsystem CPU time described in this section should be used if response times are unacceptable and if the TSS Response Time Analysis System indicates that a major portion of response time is spent in the Subsystem CPU or Other Subsystem CPU Substates.

A. PROCEDURE SUMMARY

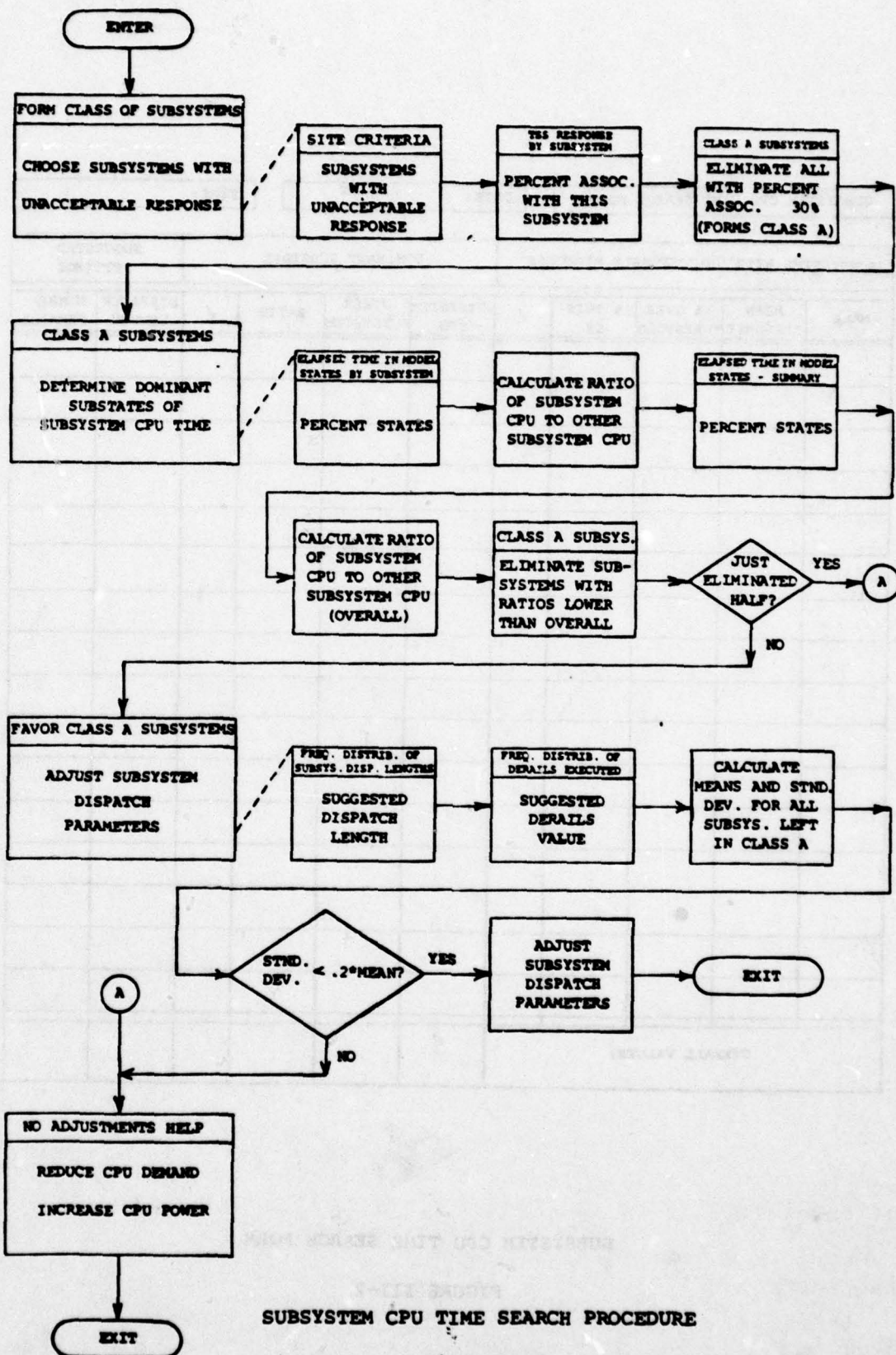
A system with unacceptable TSS response time may achieve acceptable response for most types of commands but achieve unacceptable response for a few. The commands that achieve unacceptable response may do so because certain dispatch parameters discriminate against them. The Subsystem CPU Time Search Procedure tries to adjust these parameters to achieve acceptable response on all types of commands. This adjustment usually will achieve success only if three characteristics are present: (1) many of the unacceptable responses are associated with a small number of subsystems, (2) these subsystems spend a large percentage of time in the Other Subsystem CPU substate, and (3) these subsystems have certain characteristics in common, making it possible to adjust the dispatch parameters in their favor.

Frequently, this adjustment will not be successful. Reducing the amount of CPU time needed by the TSS workload and/or increasing the CPU power available to TSS users are more likely to help. Both of these solutions are discussed later in this section. The dispatch parameter adjustment is discussed first because it involves less cost and less effort than the other two solutions.

The Subsystem CPU Time Search Procedure includes four steps: (1) Determine Subsystems with Unacceptable Response, (2) Determine Dominant Substates of Subsystem CPU Time (SS CPU or Other SS CPU), (3) Adjust Subsystem Dispatch Parameters, and (4) Reduce CPU Demand and/or Increase CPU Power. Figure III-1 charts the procedure steps.

A form (see Figure III-2) is provided with this procedure to guide and document the data collection. Several copies of the form will be required for each analysis effort.

The reports used in the Subsystem CPU Time Search Procedure are listed in Table III-1.



SUBSYSTEM CPU TIME SEARCH PROCEDURE

FIGURE III-1

TABLE III-1,
REPORTS USED IN THE SUBSYSTEM
CPU TIME SEARCH PROCEDURE

SYSTEM	REPORT
TSS Response Time Analysis System	<ol style="list-style-type: none"> 1. Elapsed Time in Model States by Subsystem 2. Elapsed Time in Model States 3. Frequency Distribution of Subsystem Dispatch Lengths (by Subsystem) 4. Frequency Distribution of Derails Executed Between CPU Eligibility Losses (by Subsystem) 5. TSS Response by Subsystem

B. DETERMINE SUBSYSTEMS WITH UNACCEPTABLE RESPONSE

The objective of this procedure step is to isolate those subsystems that are responsible for most of the unacceptable responses. Use the TSS Response Time by Subsystem report of the TSS Response Time Analysis System. Enter the data on the Subsystem CPU Time Search Form (Figure III-2). Use a separate form for each time period monitored. Note the date and time of the monitored period at the top of the form.

1. Decision: Subsystems Associated With Unacceptable Response

Using the site-specific definition of acceptable response, note which subsystems are associated with unacceptable response times. Use the Mean Response Time, Standard Deviation, and/or Percent Greater than Threshold columns of the TSS Response by Subsystem report. The columns to be used and their method of use depend on the particular definition of

acceptable response. The objective is to eliminate from consideration those subsystems that nearly always achieved acceptable response and derive a list of subsystems which account for [60%-90%] of the unacceptable responses.

2. Record Subsystem Names

- a. Report Value. The four character subsystem names are given in the Subsystem column of the TSS Response by Subsystem report.
- b. Form Entry. For each subsystem determined above to be associated with unacceptable response, enter the subsystem name in the Name column of the form.

3. Record Mean Response Time

This value and the Percent Greater than Threshold value are recorded for documentation only. An analyst may prefer to record other information depending on the criteria for deciding which subsystems were associated with unacceptable response.

- a. Report Value. The Mean Response Time, given for each subsystem, is the arithmetic average of all TSS responses with which the subsystem was associated in some way.
- b. Form Entry. Enter the value in the Mean Response column for each subsystem with an entry in the Name column.

4. Record Percent Greater Than Threshold

- a. Report Value. The Percent Greater than Threshold value represents the percentage of responses which were longer than the value listed in the Threshold column.
- b. Form Entry. Enter the value in the Percent Over Threshold column for each subsystem associated with unacceptable response.

5. Record Percent Associated With This Subsystem

These values are used in the decision step below.

- a. Report Value. The Percent Associated With This Subsystem value represents the percentage of the response time reported for a subsystem during which that subsystem was actually active. During the rest of the response time, other subsystems were active.

b. Form Entry. Enter the value in the Percent This Subsystem column for each subsystem associated with unacceptable response.

6. Choose Subsystems for Further Investigation

Subsystems that were associated with unacceptable responses should not be considered responsible for those responses if they were active only a small portion of the time.

a. Decision. Eliminate all subsystems whose Percent This Subsystem values are [$<30\%$].

b. Form Entry. Place a check in the checkmark (\checkmark) column for each subsystem not eliminated.

c. Multiple Time Periods. Repeat the six entries and decisions for each time period monitored that experienced unacceptable TSS response. Continue analysis with the next section.

C. DETERMINE DOMINANT SUBSTATES OF SUBSYSTEM CPU TIME

The objective of this procedure is to determine if the subsystems checked above cause unacceptable responses because they are denied proper CPU service. This procedure uses the Elapsed Time in Model States by Subsystem report to calculate the ratio of Subsystem CPU time to Other Subsystem CPU time. A low ratio indicates that the subsystem may benefit from a change in dispatch parameters.

1. Record Subsystem CPU Percent

For each time period with poor response and each subsystem checked above in Section III.B.6, record the percentage of time spent in the Subsystem CPU substate.

a. Report Value. Use the Percent States value for the Subsystem CPU substate. This value is listed on the Elapsed Time in Model States by Subsystem report of the TSS Response Time Analysis System. A separate report is used for each subsystem. The nearby Percent All Subsystem value should not be confused with the Percent States value.

b. Form Entry. For each subsystem checked above in Section III.B.6, enter the value from the subsystem's Elapsed Time in Model States report in the Subsystem CPU column. Insure that the data being recorded are from the same time period and date as the data already on the form.

2. Record Other Subsystem CPU Percent

a. Report Value. Use the Percent States value for the Other Subsystem CPU substate. This value is listed on the Elapsed Time in Model States by Subsystem report.

b. Form Entry. For each subsystem checked, enter the value from that subsystem's Elapsed Time in Model States report in the Other Subsystem column.

3. Record Overall Subsystem CPU Percent

For each time period monitored, record the overall Subsystem CPU Percent.

a. Report Value. For each time period, use the Percent States value for the Subsystem CPU substate from that time period's Elapsed Time in Model States Summary report.

b. Form Entry. For each time period, enter the value in the row provided at the bottom of the form, in the Subsystem CPU column.

4. Record Overall Other Subsystem CPU Percent

a. Report Value. Use the Percent States value for the Other Subsystem CPU substate from each time period's Elapsed Time in Model States Summary report.

b. Form Entry. For each time period, enter the value in the row provided at the bottom of the form, in the Other Subsystem column.

5. Calculate Ratio

a. Calculation. For each time period and subsystem, divide the Subsystem CPU value by the Other Subsystem value. For each time period, divide the Overall Subsystem CPU value by the Overall Other Subsystem value.

b. Form Entry. Enter the results in the Ratio column.

6. Decision

a. Eliminate Subsystems. Eliminate from further investigation all subsystems whose ratios are higher than [the Overall ratio for that time period].

b. Form Entry. Place a check in the checkmark (✓) column next to each subsystem not eliminated.

c. Decision. If less than [half] the subsystems checked in Section III.B.6 above were checked again in this section, skip Section III.D and proceed directly to Section III.E. Otherwise continue with Section III.D.

D. ADJUST SUBSYSTEM DISPATCH PARAMETERS

The objective of this procedure is to adjust subsystem dispatch parameters to favor those subsystems whose lack of CPU resources is causing long response times. The common dispatch characteristics (length of CPU service and number of Derails desired between losses of CPU Eligibility) of these subsystems must first be determined. If these characteristics vary widely among those subsystems responsible for long response times, no adjustment can be made that will favor them all. If these characteristics are no different than those of the other subsystems, no adjustment can be made to specially favor them.

1. Determine Subsystem Characteristics - Dispatch Length

Obtain the Frequency Distribution of Subsystem Dispatch Lengths report for each subsystem being investigated (those with a check on the form next to the Ratio column). This report shows the lengths of CPU time that a subsystem would have taken if it were granted endless, uninterruptable dispatches.

a. Suggested Dispatch Length Setting. For each such subsystem, determine the Suggested Setting for dispatch length. One of the following is advised: (1) the value below which approximately [75%] of that subsystem's dispatches lie or (2) if most of the [25%] above the first value are clustered in a single bucket, the upper limit of that bucket. The objective is to find a dispatch length that allows the checked subsystems have as long a time slice as they need, and interrupts subsystems that could use longer time slices.

b. Form Entry. Enter the Suggested Setting value for each subsystem in the Dispatch Length column.

2. Determine Subsystem Characteristics - Number of Derails

This procedure determines the number of Derails each subsystem executes between exits from the Eligible for CPU state. Obtain the Frequency Distribution of Derails Executed Between CPU Eligibility Losses report for each subsystem under investigation.

a. Suggested Number of Derails Setting. Choose the Suggested Number of Derails Setting for each subsystem using the same two rules used above for the Suggested Dispatch Length Setting.

b. Form Entry. For each subsystem, enter the Suggested Setting value in the Number of Derails column.

3. Calculate Means and Standard Deviations

a. Calculation. For each time period, calculate the mean and standard deviation of the values in the Dispatch Length column. Do the same for the Number of Derails column.

b. Form Entry. Enter the means and standard deviations at the bottom of their respective columns.

4. Decision to Change Dispatch Parameters

Standard deviations less than [0.2] times their means are considered low; otherwise, they are considered high. If one or both standard deviations are low for most time periods, modify the dispatch parameters as described in Section III.D.5. If both are high for most time periods, proceed directly to Section III.E below.

5. Changing Dispatch Parameters

This procedure involves changing one or more parameters and observing the effects on response times. The parameters to be changed are: (1) the length of CPU time allowed for each dispatch to a subsystem and (2) the number of Derails allowed before dispatch to another subsystem. If both the standard deviations used in Section 4 were low, both parameters should be changed. If only one was low, only the single corresponding parameter should be changed.

The parameters should be changed with system patches as explained in Appendix I. One parameter is .TCDEL in TSSA. The other is changed by patch number nine in Appendix I. The objective is to favor as much as possible the entire group of subsystems being investigated. If the dispatch length is to be changed, the new value should be the [next to the largest representative value recorded in the Dispatch Length column of the form]. If the number of Derails allowed is to be changed, the new value should be the [next to the largest representative value recorded in the Number Derails column of the form]. If three subsystems or less are being investigated, use the largest representative value. If the new values are the same as the old, there is nothing to change; proceed to Section III.E.

Do not lower the value of .TCDEL without installing a patch to .MFALT to restore the timer register after a MME GELBAR interval. See Appendix I patch number ten.

Monitor response times to see if they become and remain adequate. If the new response times are adequate, exit the procedure here. If they are still not adequate, go to Section III.E.

E. REDUCE CPU DEMAND AND/OR INCREASE CPU POWER

If the analyst arrives at this section, the amount of CPU time used by subsystems is probably a major contributor to long response times and a simple adjustment to dispatch parameters is not likely to help. The remaining procedures to alleviate this problem will, in most cases, be expensive and/or hard to accomplish. It, therefore, may be better to try the search procedures associated with other model substates that also consume a large part of response time. These other procedures may provide cheaper or easier ways to reduce response time.

The methods of reducing subsystem CPU time are: (1) find alternative subsystems or commands, (2) shift applications to batch, (3) discontinue wasteful practices or marginal applications, and (4) optimize user (FORTRAN or BASIC) problems that execute under TSS supervision. All of these alternatives involve close communication and cooperation with management and users. Methods of increasing effective CPU power are: (1) split a multiprocessor system into two systems with some of the users on each system, and (2) upgrade to a faster processor. These six alternatives are discussed below.

1. Find Alternative Subsystem or Commands

Another subsystem or command may achieve the same or similar results and use less CPU time. A frequently-run BASIC program recoded in FORTRAN might use less than half the CPU time. One job status query might use less CPU time than another.

If all the functions of a command or subsystem could be assumed by other commands or subsystems, the command or subsystem usually could be made unavailable with a single patch. This would insure that all users switched to the new commands or subsystems. In either case, the users must be informed and persuaded that the change is in their interest.

2. Shift Applications to Batch

Some applications might be processed as batch programs with little or no inconvenience to the user. Batch jobs would interfere less with other TSS applications (i.e., Other Subsystem substate time would be reduced) and might even be run at night. Prime batch candidates would be FORTRAN or BASIC jobs that use a relatively large amount of CPU time and are not truly interactive, i.e., the user simply enters parameters at the start and receives a series of outputs at the end.

3. Discontinue Wasteful Practices or Marginal Applications

Some practices are convenient but wasteful, e.g., turning on the DJST job status message, leaving the terminal, and returning from time to time to check on the latest status. The status of the job must be requested from GCOS every few seconds (whether it has changed or not). A VIDEO monitor in the terminal room would help eliminate this practice. The time interval between requests for status may also be increased. Marginal applications would be any applications that are not important (e.g., games) or that could be done as well manually or with a calculator.

4. Optimize User Programs

With present TSS instrumentation, it is difficult to identify user programs that should be optimized. This task would therefore involve primarily education and cooperation with the users. Some billing systems summarize terminal time and CPU time by user and session. A relatively high ratio of CPU time to session time might indicate a user

program that consumes large amounts of CPU time. A subsequent interview might determine if this indication were true and might persuade the user to optimize his program or allow his program to be optimized by an experienced programmer.

5. Split a Multiprocessor System

This solution would allow two processors to execute TSS at once, because there would be a copy of TSS in each machine. This results in: (1) possible access problems to common data, (2) memory wastage because two copies of GCOS and TSS use memory at the same time, and (3) decreased efficiency of batch applications. However, if the installation had enough memory, peripherals, front-end processors, and CPU's to split into two systems, splitting might be a relatively cheap and effective way to reduce response time.

A variation of this approach is to change to an operating system that allows more than one processor to execute TSS simultaneously. Commercial releases 3/I and after and WWMCCS 7.1 permit this.

6. Upgrade to a Faster Processor

A faster processor can execute the same subsystem in less time. It will be free to execute other subsystems sooner. Variations of this approach are: (1) add cache memory, (2) expand existing cache memory, and (3) reconfigure memory so that interlacing is possible.

7. Summary

It is impossible for this document to weight the many considerations that influence the relative desirability of the above alternatives. Rapport with users, staffing considerations, available equipment, pricing changes, and other changeable and unique factors preclude any specific recommendation among alternatives. Discussion, consultation, and thorough planning are needed when deciding how to attack the problem of long responses caused by subsystem CPU time. As stated earlier, it may be best to attempt optimization of other areas (Disk I/O Time, Memory Wait Time, etc.) first, and return to this section if these attempts are not successful.

SECTION IV. TSS EXECUTING CPU TIME SEARCH PROCEDURE

This section describes the procedures for analyzing TSS Executive CPU time. These procedures should be used if response times are unacceptable and the TSS Response Time Analysis System indicates that a major portion of response time is spent in the Executive CPU substate.

A. PROCEDURE SUMMARY

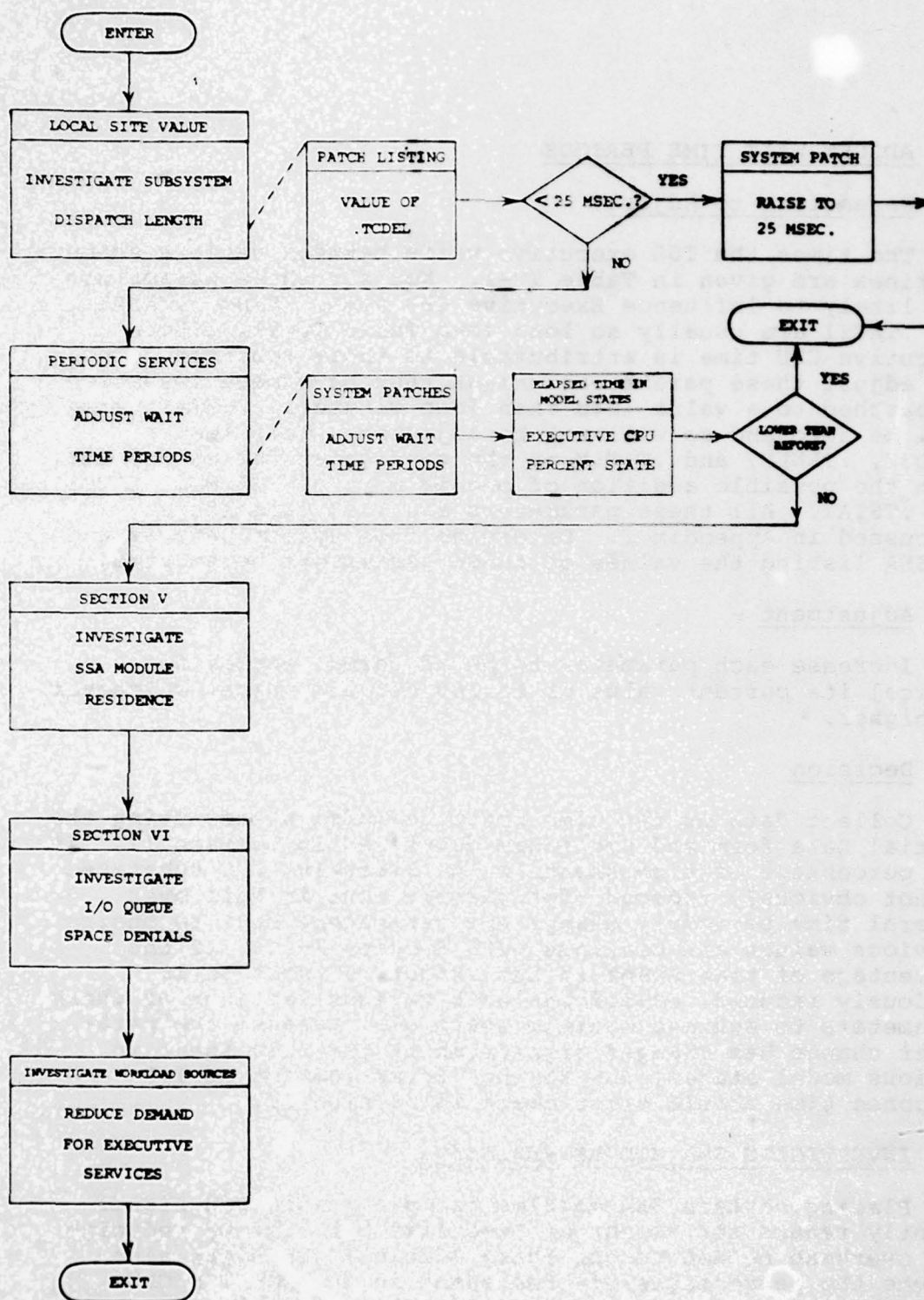
The amount of TSS Executive CPU time is controlled by several factors: the types of demands made by user subsystems, the various intervals of time waited between periodic executive functions, and the numbers of I/O Queue Space denials and Slave Service Area (SSA) module loads. There is usually only one way to ask the TSS executive for a certain type of function. Therefore, the only way to influence the types of demands made is to lower the number of demands of a particular type or types. This requires a thorough understanding of user needs and how TSS meets these needs; therefore, it is discussed last even though it may yield more results.

The procedure steps executed under the TSS Executive CPU Time Search Procedure are: (1) Adjust Subsystem Dispatch Length, (2) Adjust Wait Time Periods, (3) Investigate SSA Module Residence, (4) Investigate I/O Queue Space Denials, and (5) Reduce Demand for Executive Services. Figure IV-1 charts the procedure steps to be executed under the TSS Executive CPU Time Search Procedure. Those procedure steps requiring forms use forms from other sections of this volume.

B. ADJUST SUBSYSTEM DISPATCH LENGTH

The subsystem dispatch length, .TCDEL, is the length of CPU time given to a subsystem with each dispatch. If this parameter is set very low, it may cause high Executive CPU times.

Determine the setting of .TCDEL from system patches and a listing of TSSA (see TSSA Parameters in Appendix I). If this parameter is at the default (25 milliseconds) or higher, proceed to Section IV.C. If it is lower, raise it to the default by removing the patch or altering the code. This adjustment will probably change the amount of time spent in the various substates of the TSS Response Time Model; therefore, new Initial Data should be taken and the procedure in Section II repeated.



TSS EXECUTIVE CPU TIME SEARCH PROCEDURE

FIGURE IV-1

C. ADJUST WAIT TIME PERIODS

1. Parameters to Adjust

The times the TSS executive waits between various executive routines are given in Table IV-1. Not all these parameters are likely to influence Executive CPU time. Since .TAMRI and .TATYI are usually so long (see Table IV-1), little Executive CPU time is attributable to their routines. Do not adjust these parameters unless they have been assembled or patched to a value less than [one minute]. .TSTAT normally will be zero and so will not be adjusted. This leaves A.SD3I, .TLNLM, and .TLTLM as the candidates for adjustment, with the possible addition of one or more of .TAMRI, .TATYI, and .TSTAT. All these parameters are TSSA parameters discussed in Appendix I. Determine from system patches and a TSSA listing the values of these parameters being used.

2. Adjustment

Increase each parameter to be adjusted, either to [twice] its current value or to its default value, whichever is higher.

3. Decision

Collect data on the time spent in model states using the Initial Data form and procedure described in Section II. If the percentage of time spent in the Executive CPU substate is not obviously reduced (for example, cut in half over several time periods), change the parameters back to their previous values and continue with Section IV.D. If the percentage of time spent in the Executive Substate is obviously reduced, experiment with various settings of these parameters to achieve optimum settings. Because the parameter change has changed significantly the time spent in various model states, any further investigation of TSS response time should start again in Section II.

D. INVESTIGATE SSA MODULE RESIDENCE

Placing certain SSA modules in main memory can significantly reduce the amount of Executive CPU time by reducing the overhead needed to use these modules. This can also reduce the percentages of time spent in the Non-TSS CPU substate, because TSS is forced to relinquish the processor while a SSA module is loaded. The procedure for this investigation is contained in Section V.E. Follow that procedure and return here if placing more SSA modules in main memory is not necessary.

TABLE IV-1.
INTERVALS BETWEEN EXECUTION ROUTINE EXECUTIONS

NAME	DEFAULT VALUE	EXPLANATION
A.SD3I	1 second	Wait time between executions of routine that identifies urgent users and initiates special actions on their behalf
.TAMRI	5 minutes	Time between executions of routine to consider a memory size reduction
.TATYI	15 minutes	Interval between generation of the console status message and between swap file size reduction considerations
.TLNLM	3 seconds	Time between executing a status GEROUT to each user to detect breaks and disconnects
.TLTLM	0.5 second	Time between Line Service routine executions
.TSTAT	0.0	Time between TSRI executions. TSRI is a subsystem that periodically monitors TSS if .TSTAT is non-zero. .TSTAT should remain zero unless the TSRI statistics are being used in some worthwhile manner.

E. INVESTIGATE I/O QUEUE SPACE DENIALS

I/O Queue Space denials may also cause higher Executive CPU times. The procedure for this investigation is given in Section VI.D. Follow that procedure and return here if no change in I/O Queue space is recommended.

F. REDUCE DEMAND FOR EXECUTIVE SERVICES

The most reliable method of reducing Executive substate time is to reduce the number of requests for TSS executive services. The drawback to this method is in finding a way to reduce the number of requests. The analyst must understand how and why the users use TSS and how TSS processes their commands. Since this depends on the details of the workloads and their implementations, only suggestions can be made in this document.

Reducing demand for executive services usually means reducing the numbers of Derails executed, especially those Derails that require significant processing (i.e., Derails causing disk accesses, subsystem changes, file system module executions, etc.). Some ways this may be accomplished are: removing some applications from TSS entirely, restructuring a database to require fewer files or fewer disk accesses, and lengthening the period between some services. For example, some sites make extensive use of the DJST subsystem to report continuously the status of batch jobs. The DJST subsystem obtains the status through a Derail. In some cases, the length of time this subsystem waits between requests for job status can be increased without reducing service to the users. This increase reduces the number of swaps as well as status requests and may significantly reduce the workload on the TSS executive.

A suggested approach to reducing demand is to talk to users to find out what specific types of work TSS does and what sequences of commands are entered to cause this work to be done. The details of how TSS processes these commands should then be researched (i.e., what subsystems are used, when disk accesses are required, etc.). A more efficient way to accomplish the same work may be suggested by these details.

SECTION V. TSS WAIT FOR CPU TIME SEARCH PROCEDURE

The procedures described in this section analyze time spent while TSS is waiting for a CPU. These procedures should be used if response times are unacceptable and if the TSS Response Time Analysis System indicates that a major portion of response time is spent in the Non-TSS CPU substate.

A. PROCEDURE SUMMARY

The TSS Response Time Analysis System accumulates Non-TSS CPU time only when a user is waiting for some type of CPU service to be performed on his behalf. This means that Non-TSS CPU time represents time during which TSS needed to use a CPU and could not because other programs or GCOS functions had possession of the CPU(s).

The obvious solution to this problem is to grant TSS priority over the other jobs waiting for CPU service. This can be done by granting TSS Priority B, a well-known feature of the GCOS dispatcher.

Even if TSS has such priority, several other factors may cause TSS to wait for CPU service. If TSS's dispatch length is too short, TSS is forced to relinquish a processor before it has finished using it. It may relinquish a processor because it needs to execute an SSA module that is busy or out of core. It may be blocked from dispatch because of certain Priority B parameters. It may relinquish a processor because it has insufficient I/O queue entry space. It may be forced to relinquish a processor while certain GCOS functions are completed. Other Priority B jobs (if present) will compete with TSS for processor time. The procedures in this section investigate each of these possible problems.

This procedure includes seven steps: (1) Investigate TSS Priority, (2) Investigate TSS Dispatch Length, (3) Investigate Priority B Dispatch Blocking, (4) Investigate SSA Module Residence, (5) Investigate I/O Queue Space, (6) Investigate Line Service Interval, and (7) Investigate Intermittent Problems. Each procedure step is described in this section (see also Figure V-1).

A form (see Figure V-2) is provided with this procedure to guide and document the data collection. Each analysis effort requires a separate copy of the form.

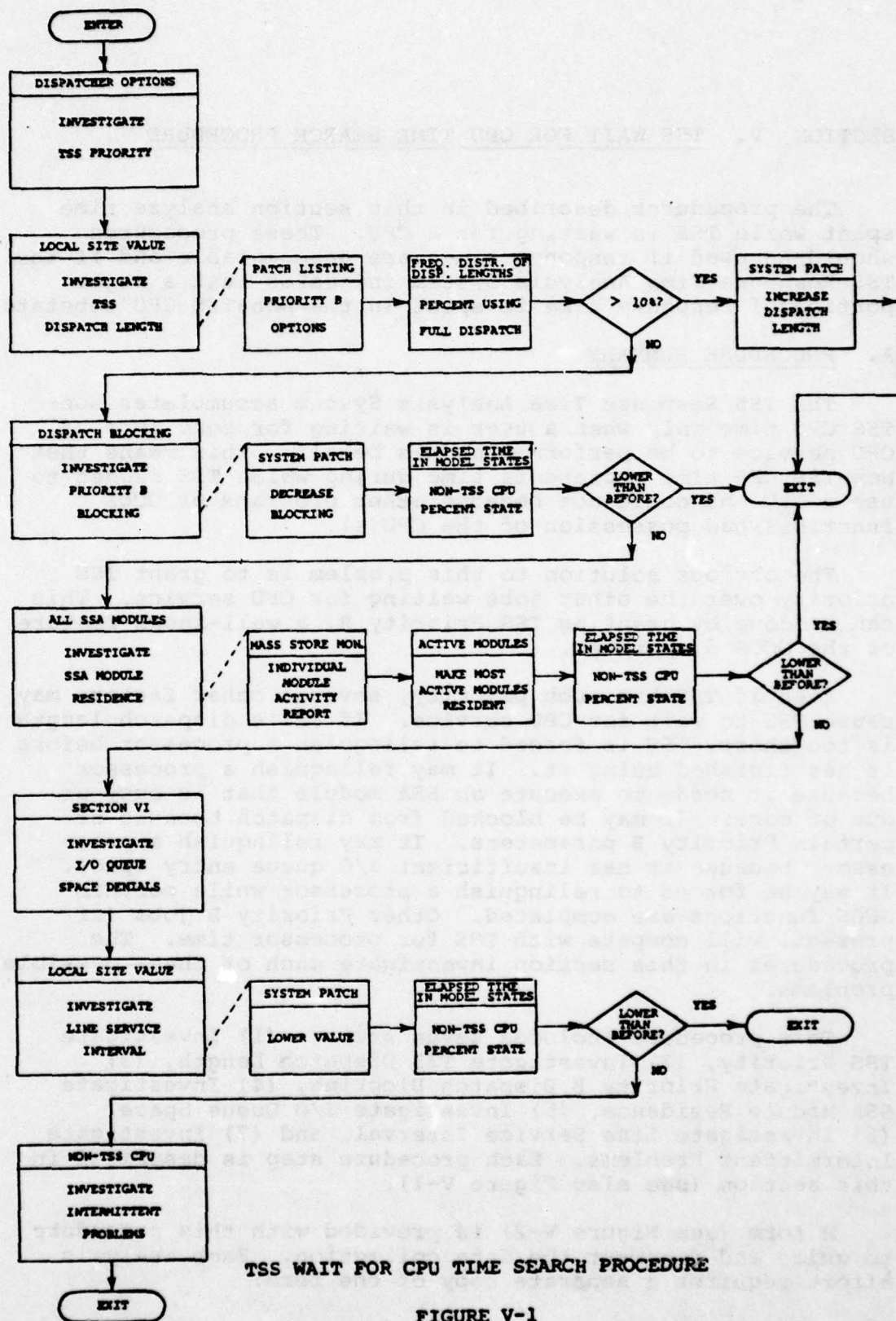


FIGURE V-1

TSS WAIT FOR CPU TIME SEARCH FORM

DATE:

[illegible]

TSS WAIT FOR CPU TIME SEARCH FORM

FIGURE V-2

The reports used in the TSS Wait For CPU Time Search Procedure are listed in Table V-1.

TABLE V-1. REPORTS USED IN THE TSS WAIT FOR CPU TIME SEARCH PROCEDURE

SYSTEM	REPORT
TSS Response Time Analysis System	1. Frequency Distribution of Dispatch Lengths 2. Elapsed Time in Model States
Mass Store Monitor	1. Individual Module Activity

B. INVESTIGATE TSS PRIORITY

The objective of this procedure step is to grant TSS the highest priority possible. If TSS already has Priority B status, proceed to Section V.C.

Most sites run with the Urgency Throughput and/or I/O Priority dispatcher options. However, these options may not help TSS compete for CPU time very well. Urgency Throughput will allow any high urgency job (including most system programs, i.e., CALC, PALC, etc.) to have equal priority with TSS. Because TSS's (CPU Time)/(I/O Channel Time) ratio may be larger than many batch jobs, selecting the I/O Priority dispatcher option may not shorten TSS response. When Urgency Throughput and I/O Priority are used together, both objections are valid. The Priority B option assures that TSS will receive high priority at all times. Priority B also makes it possible to change the length of the CPU time slice given to TSS.

Many sites are reluctant to grant TSS Priority B, fearing degradation to other jobs in the system. Experience and field tests have shown that batch jobs are rarely degraded. Granting TSS Priority B causes TSS to complete its work quickly, disconnect entirely from the CPU wait

queue, and allow other jobs full control of the CPU until TSS is given more work. Priority B for TSS may degrade other on-line jobs, and abuse of TSS priority by users may degrade performance. Users should be informed not to use commands that take inordinate processor time. This can be enforced partially by setting .TASTM in TSSA (see Appendix I) and by monitoring the CPU time consumed by each user.

Implement Priority B for TSS as shown in Appendix I. Monitor the results with the TSS Response Time Analysis System (using the Initial Data form and recording procedure in Section II) and note whether response becomes acceptable. If response is acceptable, the tuning effort may be ended. If response remains unacceptable, continue the analysis in one of two ways. If the percentage of response time associated with the Non-TSS CPU substate is still high, continue with Section V.C. Otherwise, follow the procedures in Section II to select the substate which now shows the highest percentage of response time. In any case, remove the Priority B patches only if evidence is found that other jobs are being unacceptably degraded.

C. INVESTIGATE TSS DISPATCH LENGTH

The objective of this procedure is to determine if the CPU time granted to TSS upon each dispatch should be lengthened. Unless a site has implemented the Timer Register Patch (patch 10) discussed in Appendix I, the length given to TSS is the shorter of two times: the normal dispatch length given to TSS by the dispatcher, and the dispatch length given by TSS to its subsystems (.TCDEL).

1. Determine TSS Executive Dispatch Length

a. Value. Determine the amount of CPU time allowed TSS at dispatch. If TSS has Priority B, this time is specified in the Priority B patch (see Dispatcher Parameters in Appendix I). If TSS does not have Priority B, this time will depend on the number of processors configured. It will probably be 64 milliseconds, plus 32 milliseconds times the number of processors configured.

b. Form Entry. Enter the value in the Executive Length box on the form.

2. Determine Subsystem Dispatch Length

a. Value. Determine the size of .TCDEL by examining the patch deck and a current listing of TSSA (see Appendix I for location and interpretation).

b. Form Entry. Enter the value (in milliseconds) in the Subsystem Length box on the form.

3. Determine Effective Dispatch Length

a. Decision. The amount of CPU time per dispatch that TSS can actually use is the smaller of the two dispatch lengths, unless the Timer Register patch has been implemented. In that case it is the Executive Length value.

b. Form Entry. Enter the value (depending on whether the Timer Register patch is implemented at this site) in the Effective Length box on the form.

4. Obtain Distribution of Dispatch Lengths Report

Obtain the Distribution of Dispatch Lengths report for the same period as the Initial Data currently being used. Adjust the bucket sizes before the reports are produced so that one bucket includes only the Effective Dispatch Length and a small range on either side. For example, if the Effective TSS dispatch length is 64 milliseconds, one bucket should be 60-70 milliseconds. The TSS dispatches that fall in this bucket will be those in which TSS used the entire dispatch length and was prevented from continuing by a timer runout.

5. Enter Dates

a. Report Value. Use the Start Time recorded in the upper right corner of the report.

b. Form Entry. Enter the starting date and time in the Date and Time columns on the form. Repeat this for each separate Distribution of Dispatch Lengths report.

6. Determine Percentages Using Full Dispatch

a. Report Value. Note the percentage of "Other Dispatches" falling in the bucket surrounding the Effective Dispatch Length. Few dispatches should fall in the

buckets reported below this one. If they do, recheck the Effective Dispatch Length determination to see if it was correct. A small increase in the size of the bucket (e.g., from 60-70 milliseconds to 60-75 milliseconds) may also alleviate the problem. If these steps do not help, seek professional assistance.

b. Form Entry. Enter on the form the percentage of dispatches that fell in the bucket surrounding the Effective Dispatch Length. Use the Percent Full Dispatch column. The percentage should be entered on this form for each separate Distribution of Dispatch Lengths report obtained.

7. Calculate Mean

a. Calculation. Calculate the mean (arithmetic average) of the Percent Full Dispatch values.

b. Form Entry. Enter the mean in the box at the bottom of the Percent Full Dispatch column.

8. Decision

Raise the TSS dispatch length if the mean is greater than [10%]; otherwise, proceed to Section V.D. The following paragraphs explain how to raise the effective dispatch length given to TSS.

The first method is by implementing the Timer Register patch (number 10) discussed in Appendix I. This patch will raise the Effective Dispatch Length from the subsystem dispatch length to the executive dispatch length (which is normally larger). If that patch has already been implemented or further increases are desired, the executive dispatch length should be raised.

If TSS does not have Priority B, raising its executive dispatch length is too complex to be covered in this document. If TSS does have Priority B, one can add up to 192 milliseconds to the executive dispatch length by changing the Priority B patch as explained in Appendix I.

However the Effective Dispatch Length is changed, re-execute this procedure using new data to see if it should be further lengthened. (The site should also ensure that the Priority Parameters Change time interval (.TAGPM) has been changed as recommended in Appendix I. Otherwise, it is possible that TSS will change its dispatch length back to 64 milliseconds at some point in the day.)

If response becomes acceptable, the tuning effort may be ended; otherwise, continue the analysis by recording the latest data on percentages of time spent in substates as instructed in Section II. Continue with Section V.D. if the Non-TSS CPU substate continues to have a high percentage of response time; otherwise, choose another substate and procedure as instructed in Section II. Leave the TSS Effective Dispatch Length at the new value unless no improvement was realized and other high priority jobs were impeded demonstrably by the new dispatch length.

D. INVESTIGATE PRIORITY B DISPATCH BLOCKING

In order to keep Priority B jobs from totally locking other jobs out of CPU time, GCOS periodically blocks all Priority B jobs from execution. When this blocking occurs too often, it can degrade TSS performance. The objective of this procedure step is to investigate the possibility of blocking Priority B jobs less often than before. Skip this section if TSS does not have Priority B.

Since there is no instrumentation for measuring the effects of blocking, simply change the parameters to block Priority B jobs half as often and collect new Initial Data according to Section II. If the Non-TSS CPU substate continues to have high percentages of response time (for example, not cut in half) continue with Section V.E. If the Non-TSS CPU substate shows significantly lower percentages of response time, experiment with various values of the parameter to find the point at which a higher value (less blocking) will not significantly reduce the percentage of response time spent in the Non-TSS CPU substate. Then proceed to Section V.E. if this substate's percentage is still the highest (or next to highest). Otherwise, proceed to Section II to choose another substate and procedure for investigation.

E. INVESTIGATE SSA MODULE RESIDENCE

Most sites with significant TSS usage should have some Slave Service Area (SSA) modules in main memory because: (1) those not in main memory must be loaded from disk, and if they call another module, they must be rolled out ("pushed down") to disk if the called module is not in main memory, (2) the I/O to disk consumes a certain amount of CPU overhead, and (3) no main-level processing on behalf of any user

takes place until the SSA module processing is done. This section attempts to identify which modules should be put into main memory so that they do not have to be loaded from disk each time they are called.

The present instrumentation does not record which programs used SSA modules, so it is hard to separate the SSA modules used by TSS (generally File Management System modules named .MFSxx) from those used by batch jobs. Patch number three in Appendix I causes MCOUNT (standard SSA module activity counter) to count only modules used by TSS. Use this patch, or assume that the SSA modules used most frequently by the total workload are also used the most frequently by TSS.

Obtain data on SSA module usage for several days using either MCOUNT or the Mass Storage Monitor Individual Module Activity report. If possible, restrict the data so that it reflects only periods of poor response (i.e., obtain data at the start and end of such a period and subtract to find the module usage during the period). Put the [six] SSA modules with the highest activity levels into hard core or private SSA spaces in TSS (see Appendix I, patches four and six). Hard core is preferable because other programs may use SSA modules in hard core. Another alternative is to increase the size of SSA cache. Obtain new Initial Data and note if the response time percentage spent in the Non-TSS CPU substate has dropped. If it has dropped during each period, experiment with more/fewer or a different combination of SSA modules in main memory until the best combination and number have been found. If it has not dropped in each period, take the modules out of main memory and continue to the next section.

F. INVESTIGATE I/O QUEUE SPACE

When TSS can find no I/O queue entry space for a user's I/O request, TSS relinquishes the processor. This means that TSS will get no attention from the CPU until some outstanding I/O request terminates. Because all subsystems that are eligible for CPU attention simply have to wait, they accumulate Non-TSS CPU time.

An I/O queue space problem is properly an I/O problem and tends also to elongate disk service time. Procedures for diagnosing and treating this problem appear in Section VI-D. Proceed to that section, execute the procedure, and return to Section V.G if I/O queue space was not a problem.

G. INVESTIGATE LINE SERVICE INTERVAL

The objective of this procedure is to determine if there should be a different length of time between executions of the line service routine. Many routines will store status information and wait for the next line service routine execution to pick up the status and transfer to the right routine. This is done chiefly to avoid executing too much code in Courtesy Call mode. Much of the time spent waiting for line service may be reported as Non-TSS CPU Time. Because tracing the start and end of such waits in order to separate them from Non-TSS CPU Time would be cumbersome, they are treated experimentally in this section.

Halve the line service interval (.TLTLM) as explained in Appendix I and obtain Initial Data (see Section II) on the changed system. Compare the percentages of time spent in the Non-TSS CPU substate. If Non-TSS CPU time has decreased in each instance, experiment with different settings of the line service interval. Note that an excessively low setting will increase Executive CPU time significantly. If Non-TSS CPU time has not decreased during every period of poor response, return the line service interval to its original setting and proceed to Section V.H.

H. INVESTIGATE INTERMITTENT PROBLEMS

Several conditions may cause intermittent poor response time in a way that causes more time to be spent in the Non-TSS CPU substate. Since no tool collects data on these conditions and they involve management aspects as well as technical considerations, definite procedures concerning them do not exist. They are mentioned here in an attempt to give all possible guidance in case the other solutions in this section fail to eliminate all response problems. Solutions to these problems depend on local conditions (except perhaps the swap file space problem, which can nearly always be solved easily and cheaply by increasing the minimum size); only suggestions are made.

The conditions discussed here are mainly interactions with the operating system where conditions outside TSS temporarily prevent TSS from continuing. If TSS tries to write to the accounting file while the accounting tape is experiencing errors or is being changed, TSS waits until the problem is resolved. The same situation occurs if TSS attempts to write its status message to the console and the console is down or so busy the message cannot be queued.

The solutions to these problems are to keep the accounting file and console error-free and to avoid writing unnecessary information to either. In particular, do not write type 14 or type 19 accounting records from TSS (see .TSTAT and .TSSAS in Appendix I) unless the information definitely is required.

Frequent memory size changes or swap file size changes can cause response time problems. System loading and/or WWMCCS security requirements may cause TSS response to be delayed noticeably each time a memory size or swap file size change occurs. Lengthen the size reduction interval (.TAMRI) or raise the minimum TSS memory size (.TASMS) during prime usage periods to cut down on the number of size changes. Raise the minimum swap file size, as discussed in Section VII.C, to cut the number of swap file size changes.

When TSS detects nothing to do, it "goes to sleep" via a MME GEWAKE for .TAGMI (default of 14) seconds. At some sites, TSS may not always be "awakened" immediately if some process ends during that interval. If this happens, a period of up to .TAGMI seconds may be added to the response concerned. This extra time can be reduced by lowering .TAGMI, although TSS overhead will be somewhat greater.

If TSS is not the only Priority B job at this site, other Priority B jobs probably will interfere with good TSS performance periodically. Raising TSS dispatch length and lowering its frequency count (see Appendix I) will help somewhat. Optimization and/or careful analysis of the execution characteristics of the other job may also help. A site may want to experiment with making the other job a non-priority job and using Urgency Throughput or I/O Priority dispatch options to try to elevate it over the rest of the workload.

SECTION VI. DISK I/O TIME SEARCH PROCEDURE

This section describes the procedures for analyzing the time spent waiting for disk I/O to complete. These procedures should be used if response times are unacceptable and the TSS Response Time Analysis System indicates that a major portion of response time is spent in the Subsystem Disk I/O state.

A. PROCEDURE SUMMARY

High Disk I/O times may be caused by a general disk I/O problem (such as queuing for channels or devices), by wrong usage or placement of TSS work files, by insufficient I/O queue space in TSS, and by excessive numbers of disk accesses. The general disk I/O problems are referred to the disk-related tests of the Batch Turnaround Time Analysis Procedures. The usage and placement of TSS work files is addressed as part of these tests. Occurrences of insufficient I/O queue space are counted and extra space added, if needed. Suggestions are given to reduce the numbers of disk accesses.

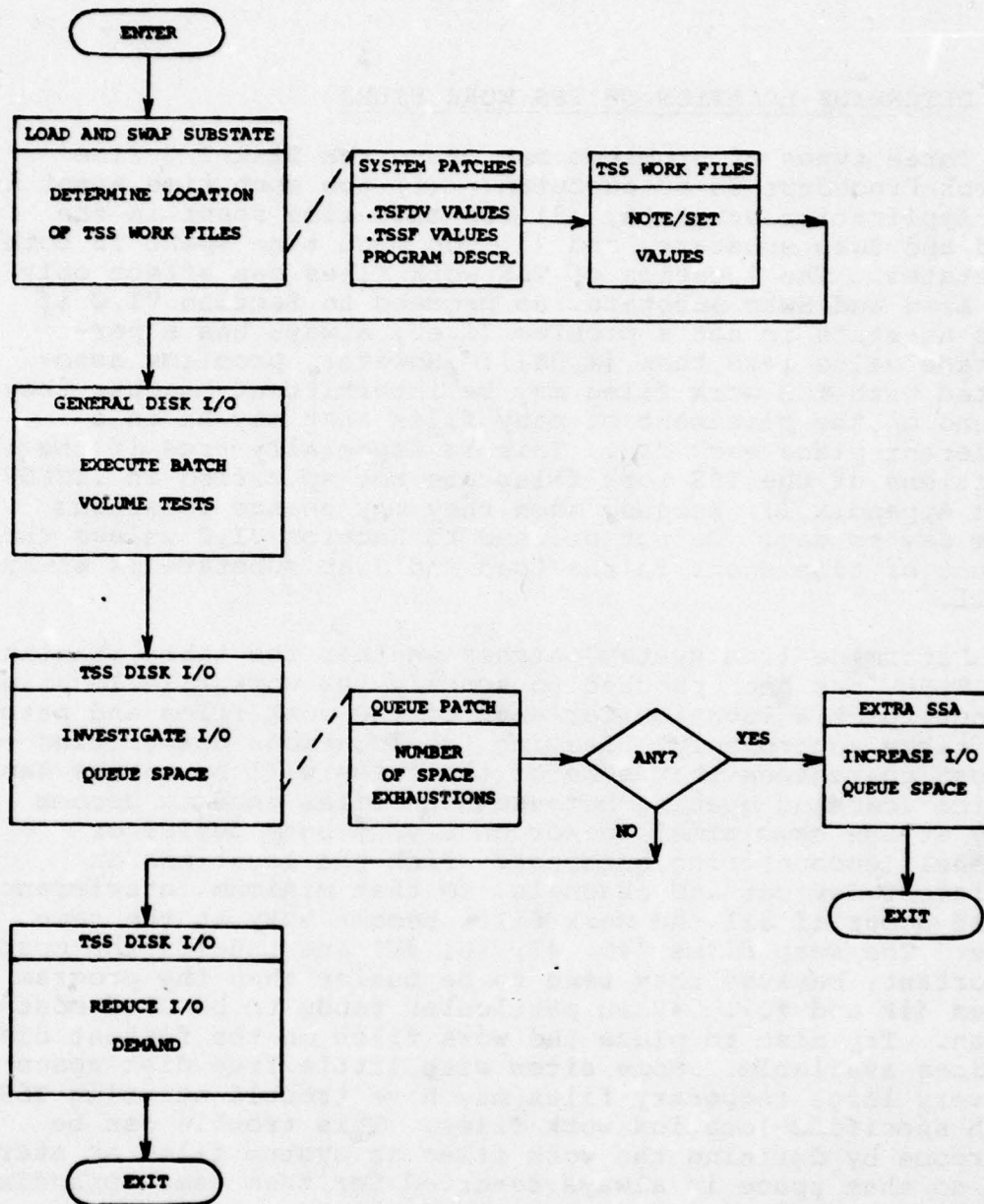
This procedure includes four steps: (1) Determine Placement of TSS Work Files, (2) Execute Batch Volume Tests, (3) Investigate I/O Queue Space, and (4) Reduce I/O Demand. Each procedure step is described in this section. Figure VI-1 charts the procedure steps that analyze Disk I/O Time.

The reports used in the Disk I/O Time Search Procedure are listed in Table VI-1.

TABLE VI-1.

REPORTS USED IN THE DISK I/O TIME SEARCH PROCEDURE

SYSTEM	REPORT
TSS Response Time Analysis System	1. Elapsed Time Spent in Model States
	2. Elapsed Time Spent in Model States by Subsystem
TSTAT	1. Status



DISK I/O TIME SEARCH PROCEDURE

FIGURE VI-1

B. DETERMINE LOCATION OF TSS WORK FILES

Three types of problems may cause the Disk I/O Time Search Procedure to be executed: (1) too much time spent in the Application substate, (2) too much time spent in the Load and Swap substate, and (3) too much time spent in both substates. The location of TSS work files can affect only the Load and Swap substate, so proceed to Section VI.C if this substate is not a problem (i.e., always has a percentage value less than [1.0%]). However, problems associated with TSS work files may be intermittent because they depend on the placement of many files that may be in a different place each day. This is especially true if the locations of the TSS work files are not specified in .TSFDV (see Appendix I), because then they may change locations from day to day. Do not proceed to Section VI.C unless the amount of time spent in the Load and Swap substate is always small.

Determine from system patches whether the table starting at .TSFDV has been patched to specify the work file locations. If not, pick a location for each of the work files and patch the table accordingly. Leaving the locations unspecified almost guarantees that some of the files will be on the same device (causing queuing because both files tend to become busy at the same time) and/or on a very busy device or channel (encountering queuing). Pick the locations on different devices and channels, so that minimum interference would occur if all the work files become busy at the same time. The swap files (#S, #T, #U, #V) are usually the most important, because they tend to be busier than the program files (#P and #Q). #V in particular tends to be used most often. Try also to place the work files on the fastest disk devices available. Some sites with little free disk space or very large temporary files may have trouble starting TSS with specified-location work files. This trouble can be overcome by defining the work files as system files at start-up, so that space is always reserved for them (see Appendix I).

Note the locations of the work files, whether fixed previously or as a result of the above discussion. Keep these locations in mind when executing the tests in Section VI.C below. If a file relocation is recommended by the tests, give the TSS work files preferential treatment. Depending on the number of TSS users and the relative importance of TSS at a site, it may be justifiable to keep some disk devices or channels relatively idle, locate the work files on these devices/channels, and so ensure minimum queuing delay when accessing the work files.

.TSSF should also be patched so that the maximum number (4) of swap files are used. The swap files are relatively small, so that little disk space would be saved by using fewer swap files. A larger number of swap files enables TSS to swap more subsystems at a time.

In heavily used systems, two or more users often may be trying to load subsystems from the program files (#P and #Q) at the same time. Most of these systems will experience heavy queuing for #P and little activity on #Q, because each subsystem resides on #P unless TSS is patched to request #Q for that subsystem. This problem has two solutions. The first solution is to put enough subsystems on #Q to even out the usage. The second solution is to put the highly used subsystems on both #P and #Q, so TSS can choose the file that is most idle at the time the load is needed. The second solution does not even the usage between #P and #Q because TSS will load from #P if both files are idle. A combination of the two solutions may be used.

The information needed for this change is given in the TSTAT output. The number of loads of each subsystem is given in the "# Times Executed" column. Appendix I (.TPD) explains how a subsystem's program descriptor chooses the program file the subsystem will use.

C. EXECUTE BATCH VOLUME TESTS

Execute the Pathway Utilizations, Seek Elongation, and Device Errors tests from the Batch Turnaround Time Analysis Procedures. Note that the tuning steps for the Pathway Utilizations Test may erase gains previously made by executing the Seek Elongation Test. To some extent, the reverse also happens. It may be advisable to execute the Pathway Utilizations Test first and then to execute it again following execution of the Seek Elongation Test if significant numbers of files are moved.

If no significant tuning steps are taken as a result of these tests, proceed to Section VI.D. If significant tuning steps are taken, gather another Initial Data set according to Section II. Continue to Section VI.D unless the percentage of response time spent in the Subsystem Disk I/O state has dropped significantly and consistently (every time period significantly lower). In this case, return to Section II to choose a substate and procedure.

D. INVESTIGATE I/O QUEUE SPACE

Every I/O to disk or terminal requires I/O queue space. When TSS cannot find space for a new I/O to start, it simply relinquishes the CPU and waits for an ongoing I/O to finish. Meanwhile, the requested I/O and all users waiting for CPU processing simply wait. This type of situation is sufficiently injurious that it should never be allowed to happen.

Apply the I/O Queue patch (Number 1) from Appendix I to count the number of TSS relinquishes of the CPU due to insufficient I/O queue space. This patch will not count all such relinquishes, but should count some if insufficient I/O queue space is a problem. The patch accumulates counts of relinquishes into a word within TSS. The contents of this word must be examined before TSS is terminated to see if its contents have changed. Any change signifies that I/O queue space has been exhausted at some point since TSS was initiated.

Use the patch for several days or for several periods of poor response, whichever is longer. If no counts are ever accumulated, proceed to Section VI.E. Otherwise, take the tuning steps below.

The TSS parameter limiting the number of concurrent users also governs the amount of I/O queue entry space. A small increase in this parameter (i.e., one additional concurrent user) will often provide an additional SSA, part of which is used for I/O queue space. The number of SSA's for the various numbers of concurrent users allowed is given in Table VI-2. Ascertain the current maximum number of users (see .TFMAX in Appendix I) by checking system patches and a listing of TSSA. To increase the amount of I/O queue space, .TFMAX should be increased enough to cause an additional SSA to be allocated.

Increase .TFMAX if TSS relinquish counts were observed. Verify that an extra SSA (extra 1K below the LAL) is being allocated. If not, .MPOPA should be patched (see Appendix I patch number 5) to allocate more SSA's to TSS initially.

TABLE IV-2. MAXIMUM TSS USERS VERSUS NUMBER OF SSA'S

MAXIMUM NUMBER OF USERS PARAMETER SETTING	NUMBER OF TSS SSA'S
1-18	3
19-36	4
37-54	5
55-72	6
73-90	7
$18(n-1)+1$ to $18n$	$n+2$

E. REDUCE I/O DEMAND

If the above sections fail to find a disk I/O problem, it may be that the large amount of time spent in the Disk I/O state of the TSS Response Time Model is simply due to large numbers of I/O requests to disk. It may be possible to reduce the number of I/O requests. The subsystems responsible for the most I/O requests may be isolated with the Elapsed Time in Model States by Subsystem report of the TSS Response Time Analysis System. Subsequent interviews with these subsystems' users could determine why these subsystems are used and how the number of I/O requests could be reduced. Some suggestions are:

1. Programmers who continually run the same large program should be encouraged to compile it and run it using the object code.
2. If one small section of a large file is changed often, it should be made a separate file. Otherwise, the entire file has to be read at least twice to make the change. When the entire file is to be used, the two parts can be merged or concatenated by using certain features of the RUN, OLD, and SAVE commands.

3. Periodic status subsystems that are responsible for large numbers of disk I/O requests (see the Number of Entries for the Disk I/O state on the subsystem's Elapsed Time in Model States by Subsystem report) should have their periods increased. Periodic status subsystems are those subsystems (such as DJST) that repeatedly ascertain the status of some part of the system. By increasing the period (so the status is ascertained less often), fewer disk I/O's would be required.

SECTION VII. MEMORY WAIT TIME SEARCH PROCEDURE

This section describes the procedures for analyzing time spent in the Memory Wait model state. These procedures should be used if response times are unacceptable and if the TSS Response Time Analysis System indicates that a major portion of response time is Memory Wait time.

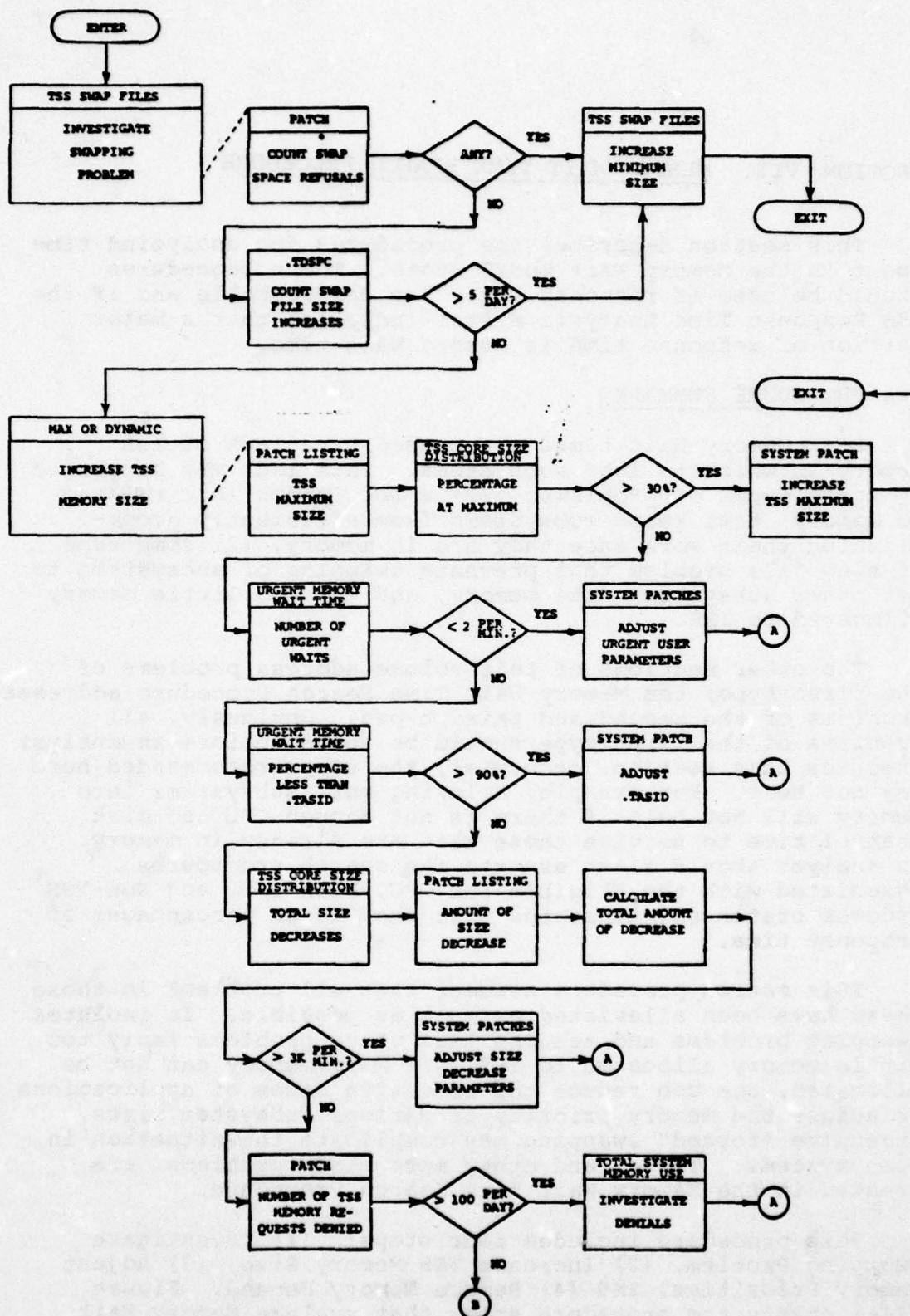
A. PROCEDURE SUMMARY

Long Memory Wait times are caused by a lack of TSS memory in which to load subsystems. This lack may be caused by three types of problems: (1) a bottleneck (not related to memory) that keeps subsystems from efficiently accomplishing their work once they are in memory, (2) some type of swap file problem that prevents swapping of subsystems to let other subsystems into memory, and (3) too little memory allocated to TSS.

The other sections of this volume address problems of the first type; the Memory Wait Time Search Procedure addresses problems of the second and third types. Obviously, all problems of the first type should be solved before an analyst executes this section; otherwise, the steps recommended here may not help. For example, allowing more subsystems into memory will not help if there is not enough CPU and disk channel time to service those that are already in memory. An analyst should first execute the search procedures associated with the Eligible for CPU, Disk I/O, and Non-TSS Process states and substates that show large percentages of response time.

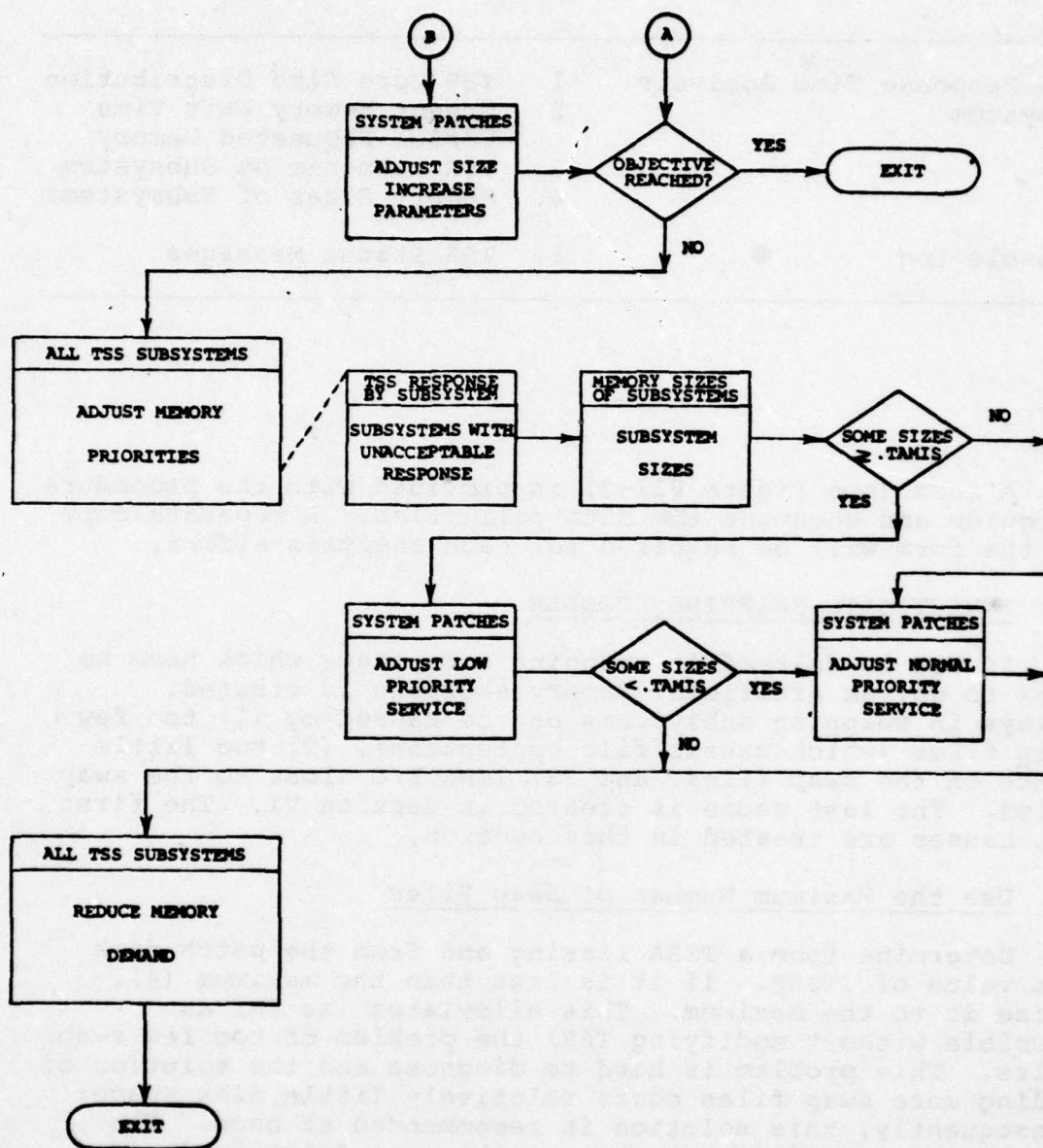
This search procedure assumes that all problems in those areas have been alleviated as much as possible. It isolates swapping problems and assumes that other problems imply too little memory allocated to TSS. If more memory can not be allocated, one can reduce the effective sizes of applications or adjust the memory priority of various subsystem sizes. Excessive "forced" swapping may complicate the situation in some systems. These, and other more minor problems, are treated in the Memory Wait Time Search Procedure.

This procedure includes four steps: (1) Investigate Swapping Problem, (2) Increase TSS Memory Size, (3) Adjust Memory Priorities, and (4) Reduce Memory Demand. Figure VII-1 charts the procedure steps that analyze Memory Wait Time. Table VII-1 lists the reports used in this section..



MEMORY WAIT TIME SEARCH PROCEDURE

FIGURE VII-1



MEMORY WAIT TIME SEARCH PROCEDURE

FIGURE VII-1 (Cont.)

TABLE VII-1. REPORTS USED UN THE MEMORY WAIT TIME SEACH PROCEDURE

TSS Response Time Analysis System	<ol style="list-style-type: none"> 1. TSS Core Size Distribution 2. Urgent Memory Wait Time Versus Requested Memory 3. TSS Response by Subsystem 4. Memory Sizes of Subsystems
Console Log	<ol style="list-style-type: none"> 1. TSS Status Messages

A form (see Figure VII-2) is provided with the procedure to guide and document the data collection. A separate copy of the form will be required for each analysis effort.

B. INVESTIGATE SWAPPING PROBLEM

If TSS is delayed in swapping subsystems which have no work to do, an artificial memory shortage is created. Delays in swapping subsystems can be caused by (1) too few swap files (which causes file contention), (2) too little space on the swap files, and (3) long I/O times to the swap files. The last cause is treated in Section VI. The first two causes are treated in this section.

1. Use the Maximum Number of Swap Files

Determine from a TSSA listing and from the patch deck the value of .TSSF. If it is less than the maximum (4), raise it to the maximum. This alleviates (as far as possible without modifying TSS) the problem of too few swap files. This problem is hard to diagnose and the solution of adding more swap files costs relatively little disk space; consequently, this solution is recommended at once. The problem of insufficient space on the swap files is decided by measurement.

2. Measure Swap File Size Changes

TSS swap files are dynamic. They can change size at any time if TSS decides there is a need to add or delete space. However, this feature brings attendant problems: (1) the

TSS executive is engaged during a size change and cannot service users (various factors may cause delays during this process, so that users are denied service for a significant time) and (2) extra space to add to a swap file may not always be available. For these reasons it is advisable to keep size changes to a minimum. The patch discussed below counts the number of times the extra space was not available, and the TSS executive itself already keeps count of the size changes.

a. Swap Space Refusal Value. Use the patch (number 7) discussed in Appendix I to count swap space refusals.

(1) Location of Value. At the end of each day, use the PEEK console verb to examine the cell into which the patch accumulates swap space refusals. This cell may change from site to site and is chosen by the person applying the patch.

(2) Form Entry. Enter the value and the date in the Swap Space Refusals and Date columns, respectively.

b. Swap File Size Increases.

(1) Location of Value. The TSS executive counts swap file size increases in the upper half (bits 0-17) of .TDSPC in TSSA. To find the value, use the PEEK console verb at the same time as above. Note that this value is reset to zero only when TSS is TERM'ed and restarted.

(2) Form Entry. Enter the value in the Size Increases column.

c. Decision. If there were [no] swap space refusals and fewer than [5] size increases per day, proceed to Section VII.C. Otherwise, increase the minimum swap file size as described below.

d. The New Parameter Setting. The parameter most likely to alleviate the problem is the minimum swap file size. A large setting for this parameter assures that TSS will usually have enough swap file space for most needs without requesting additional space from GCOS. This avoids the overhead, delays, and possible denials.

Since the figures kept by TSS on swap file space needs are incomplete, a good candidate parameter setting can only be estimated. The following calculations provide a method for making the estimate. Using the TSTAT Program Load and Swap in Sizes histogram, obtain the average subsystem swap/load size, as follows:

(1) Calculation. Calculate the average subsystem load size. Multiply each Number of Occurrences value in the left column by the Maximum Size value opposite it in the right column, add the results, and divide by the Total value. The result will be slightly larger than the average subsystem load size.

(2) Form Entry. Enter this size in the Mean Subsystem Size column on the form (Figure VII-2). Enter the date of the data in the Date column.

(3) Report Value. The Largest Number of Users value, on the first page of the TSTAT output, gives the maximum number of simultaneous users.

(4) Form Entry. Enter the value in the Maximum Users column. Repeat these four steps for each date. Note that this data comes from internal TSS data that is never reset. TSS must be TERM'ed and restarted each day for the data to be valid.

(5) Calculations. Calculate the means of the data in each column.

(6) Form Entry. Enter each mean at the bottom of its column.

(7) Calculation. The new parameter setting (expressed in "links" or 320-word blocks) is given by the formula:

$$\text{New} = \text{MSS} * 3.2 * \text{Max} \div 2.0 = 1.6 * \text{MSS} * \text{Max}$$

where MSS is the Mean Subsystem Size in 1024 word blocks, 3.2 converts from 1024 word blocks to 320 word blocks, Max is the mean Maximum Users number, and 2.0 compensates for multiple swap files. (The

2.0 would be 4.0 except for allowances for data bias and swap file space fragmentation.) The patch to .TSFS will cause each swap file to use this figure as its minimum size.

e. Increase Swap File Space. Change the minimum swap file size as described above. Monitor swap space refusals and the contents of .TDSPC for several days. Make further increases to .TSFS if swap space refusals continue or if the number of increases does not drop to less than [5] per day.

f. Follow-up. If Memory Wait Time continues to be high compared to other states and substates, continue this procedure at the next section. Otherwise, return to Section II and choose another substate and procedure.

C. INCREASE TSS MEMORY SIZE

The objective of this procedure is to increase the memory available for TSS subsystems, and, thereby, to shorten memory waits. Increasing TSS memory size can occur in three basic ways. If TSS spends most of its time at its self-imposed maximum size, that size could be raised. If it seldom reaches that limit, it may be that GCOS denies TSS core increase requests or that TSS does not recognize the need to request an increase. The second and third ways to increase TSS memory size are, therefore, to make GCOS core more available and to induce TSS to request memory more often. The third way concerns the parameters and variables TSS uses to decide (1) whether a user has waited long enough to be considered "urgent" and (2) what special actions are to be taken on his behalf, such as requesting more memory from GCOS.

1. Investigate Dependence on Maximum Memory Size

This step determines whether to increase the maximum TSS memory size.

a. Determine Maximum Size. Determine the TSS maximum memory size from system patches and a listing of TSSA. Since this can be dynamically altered from the console, determine (from the console log, if necessary) whether operators ever change it. If they do, determine what it is at the various times during the shift.

(1) Report Values. The location for any patch of maximum TSS size is .TAMMS (see Appendix I). The maximum TSS size is reported at the console in the first line of the TSS console status message.

(2) Form Entry. Record the maximum size in the Maximum Size column. If this size varies, make a separate entry each time the value changes.

b. Determine Actual Size. Determine what sizes TSS actually assumes during the data collection periods. If the maximum size is constant over each period of data collection, obtain the TSS Core Size Distribution Report; otherwise, use the TSS status messages from the console log.

(1) Calculation. Determine what percentage of time TSS is at its maximum size.

(2) Form Entry. Enter this figure on the form in the Percentage at Maximum column.

(3) Calculation. Average the values in the Percentage at Maximum column.

(4) Form Entry. Enter the mean value in the labeled box at the bottom of the column.

c. Decision. If the mean Percentage at Maximum value is greater than [30%], or if most of the unacceptable response is experienced while TSS is at its maximum core size, experiment with increased maximum TSS core sizes (Section VII.C.2). Otherwise, try to increase TSS size beneath the maximum by proceeding to Section VII.C.3. If the figure is close to [30%], an analyst may want to execute both sections.

2. Increase Maximum Memory Size

Raise the maximum size of TSS memory (.TAMMS). Each site must decide for itself how much to raise the parameter. The number and type of considerations that affect this decision preclude a specification of an amount to add. If possible, try to add enough that TSS rarely reaches the maximum size. This could be attempted for a short period or periods to show whether the increased memory size clearly reduces response times. Experiment with various sizes to

find one at which all the trade-offs (response times, available memory for the rest of GCOS, etc.) are equal. If raising the TSS maximum size does not demonstrably reduce response times, reset it to its previous value and proceed to Section VII.D. If response times are reduced, proceed to Section II and choose a new substate and procedure (perhaps this one) based on new Initial Data. If response times are reduced enough, the tuning effort may be ended.

3. Increase TSS Size Beneath the Maximum

If TSS size is too small but the TSS maximum size is rarely reached, one of the following is true: (1) TSS cannot obtain additional memory, (2) TSS rarely classifies users as urgent, (3) TSS rarely tries to increase memory on behalf of an urgent user, or (4) TSS decreases its memory too much. This section attempts to find which of these four possibilities is true. More than one possibility may be true. Note that the site may simply wish to raise the minimum size of TSS (.TASMS). This would eliminate the need to distinguish between the four possibilities and would directly cause TSS to increase its size. It will, however, cause TSS to remain at a high memory size during periods of relative idleness, thus wasting some memory. If the solutions proposed in this section fail to work, the site may wish to raise .TASMS depending on the priority of TSS and the availability of memory during periods of TSS idleness.

Obtain two reports of the TSS Response Time Analysis System--the TSS Core Size Distribution report and the Urgent Memory Wait Time Versus Requested Memory report. Note that certain parameters are required to be input to the data reduction for the latter report to be meaningful (see Appendix H for the parameter names). These may be obtained by consulting the patches made to TSS (patch edit and startup deck), a TSSA listing, and Appendix I. In addition, use a threshold value equal to the value of .TASID (see the same sources).

a. Date and Session Length

(1) Report Values. The Start Time, given in the upper right corner of the report, gives the date and time data collection was started. The Session Length, in the same corner, gives the length of the data collection in seconds.

(2) Form Entry. Enter the date, time, and length in the Date/Time and Length columns in the part of the form labeled Increase TSS Size Beneath Maximum.

b. Number of Urgent Waits

(1) Calculation. Total the Number of Waits column from the Urgent Memory Wait Time versus Requested Memory report.

(2) Form Entry. Enter the total on the form in the Urgent Waits column.

c. Percentage Less Than .TASID.

(1) Calculation. For each Urgent Memory Wait Time versus Requested Memory report, calculate the mean Percent Less Than Threshold value for all the rows (all memory sizes) in that report in the following way: For each row with a non-zero Percentage Less Than Threshold value, multiply that value times the Number of Waits value for that row. Write the result in a column next to the Percentage column on the report, as shown in Figure VII-3. Total this new column (total is four in the figure) and divide by the total (20 in the figure) for the Number of Waits column calculated above in Section VII.C.3.b.(1).

(2) Form Entry. Enter the result as a percentage in the Percentage Less Than .TASID column on the form. In the figure, this percentage is $4 \div 20 = 20\%$.

d. Size Decreases

(1) Report Value. The total number of size decreases is given at the bottom of the TSS Core Size Distribution report.

(2) Form Entry. Enter the value in the Decreases column.

e. Amount of Decrease

(1) Value. Consult TSS patches and a listing of TSSA to obtain the value for the amount of memory that TSS relinquishes each time it decreases its size. This is the value of .TASRI (see Appendix I).

REPORT 12

TAPE 0'S 24501, 24504, 24505

SESSION LENGTH (SECS) 1471.42
 START TIME 120777 1542.1642
 STOP TIME 120777 1546.6928
 ISS ACTIVITY FOR 1471.42 SECS

URGENT MEMORY WAIT TIME VS. REQUESTED MEMORY

SIZE REQUESTED	WAIT TIME (SECS)	PCT	NUMBER OF WAITS	MEAN WAIT (SECS)	STD DEV (SECS)	THRESHOLD (SECS)	PCT LESS THAN THR	MIN VAL (SECS)	MAX VAL (SECS)
1K	0.	0.	0	0.	0.	1000.0	0.	0.	0.
2K	2.9	2.41	2	1.63	0.551	1000.0	50.00	975.9	1977.9
3- 4K	2.0	1.67	1	1.98	0.	1000.0	0.	1981.6	1981.6
5- 6K	2.7	2.24	2	1.35	1.273	1000.0	50.00	80.9	2626.0
7- 8K	2.0	1.67	1	1.98	0.	1000.0	0.	1982.4	1982.4
9- 11K	0.	0.	0	0.	0.	1000.0	0.	0.	0.
12- 15K	12.5	10.53	3	6.17	2.894	1000.0	33.33	574.6	7751.0
16- 20K	0.	0.	0	0.	0.	1000.0	0.	0.	0.
21- 26K	45.3	18.68	7	5.54	4.797	1000.0	14.29	294.3	14607.9
27- 36K	45.9	39.64	3	15.29	10.103	1000.0	0.	1954.3	26230.7
37- 50K	0.	0.	0	0.	0.	1000.0	0.	0.	0.
OVER 50K	4.9	4.14	1	4.90	0.	1000.0	0.	6994.1	6994.1
			<u>20</u>				<u>4</u>		

SAMPLE CALCULATION OF THE
 PERCENTAGE LESS THAN .TASID

FIGURE VII-3

(2) Form Entry. Enter the value (in 1024 word blocks) on the form in the box labeled .TASRI.

(3) Calculation. For each data period, multiply the number of decreases by the value of .TASRI (expressed in 1024 word blocks).

(4) Form Entry. Enter the products in the Memory Released column on the form.

f. Average Values

(1) Calculation. Average each column just completed except the Date/Time and Decreases columns.

(2) Form Entry. Enter each average at the bottom of its column.

g. Decision. If the average Urgent Waits figure is less than the average session length (expressed in seconds) divided by [30], assume that TSS rarely classifies users as urgent and adjust the urgent user parameters as described in Section VII.C.2.h. If the Percentage Less Than .TASID figure is greater than [90%], assume that TSS rarely tries to increase memory on behalf of an urgent user and adjust .TASID as explained in Section VII.C.2.i. If the average Memory Released value is more than the session length (in seconds) divided by [20] (15K per five minutes of session length), assume TSS decreases its memory too much and adjust the TSS size decrease parameters as described in Section VII.C.2.j. If none of the above conditions is true, assume that GCOS may be denying TSS requests for additional memory and proceed to Section VII.C.2.k below. After making the recommended adjustments, collect new data during periods of high memory wait. Fixing one problem may cause another to become evident. Follow steps a. through g. once again.

h. Adjust Urgent User Parameters. If the user has reached this section, Memory Wait times are high but few users are classified as urgent users. The parameters involved in this classification are listed in Table VII-2. The "Desired Direction of Change" provides the direction in which the parameter should be changed to cause more users to be classified as urgent. This section describes how to choose candidate parameter settings. Specific settings are not recommended here.

TABLE VII-2.
URGENT USER CLASSIFICATION PARAMETERS

SYMBOLIC PARAMETER NAME	DEFAULT VALUE	DESIRED DIRECTION OF CHANGE	DESCRIPTION
A.SD3I	1 Second	Lower	Urgent user classification is performed every A.SD3I seconds.
.TASWT	3 Seconds	Lower	User must wait at least .TASWT seconds before he can be classified urgent.
.TASWF	4K Words/Second	Higher	User's subsystem size is divided by .TASWF to determine how many seconds he must wait before he is classified urgent. (A 16K subsystem would wait 4 seconds.)
.TAMIS	36K Words	Higher	Any program above .TAMIS must wait .TALPP times as long as he normally would before he is classified urgent (36K means a wait of 36 seconds).
.TALPP	4	Lower	See .TAMIS description.

The first parameter, A.SD3I, limits the execution of the code that (1) identifies urgent users and (2) tries to increase memory on their behalf. This parameter should be smaller than the average response desired; otherwise, it would be possible for a response to be so long that it would be unacceptable even before TSS identified it as needing the extra priority of being an "urgent" user request. A.SD3I should not be smaller than [.5] of both .TASWT and .TASID (Default = 1 second); the status of urgent users would otherwise be taken two or more times before any change occurred. The default value should usually work well.

A user cannot be considered urgent until he has waited .TASWT seconds. If the site wishes to guarantee a certain response time, .TASWT must be below that response time; otherwise, users will be able to exceed the desired response, not yet be in memory, and TSS will not recognize that there is any reason to give them special treatment. However, .TASWT should not be set so low that most memory waits become urgent waits. This will tend to make TSS use more main memory than it really needs, and may even cause excessive swapping if A.MTQ is low (see Appendix I).

Once a user has waited .TASWT seconds, .TASWF controls the additional time (if any) the user must wait to be classified urgent. The core size of the user's subsystem (in 512-word blocks) is divided by .TASWF (default of 8). The result is the number of seconds the user must have waited to be classified urgent. Using default values, a 12K subsystem must therefore have waited three seconds; a 16K subsystem must have waited four seconds. An 8K subsystem must still have waited three seconds, because all subsystems must have waited .TASWT seconds. .TASWF must be large enough that users can be declared urgent before they have waited so long that their response will be unacceptable. If .TASWF is very large (e.g., 36K per second), it allows all subsystems of all sizes to become urgent upon waiting .TASWT seconds. This is not necessarily good or bad; it depends on the relative priority the site wishes to afford larger subsystems.

.TAMIS and .TALPP exist to penalize subsystems that are considered at the upper limits of memory size. If a subsystem is .TAMIS or over in size, .TALPP is multiplied by the normal amount of time (calculated using .TASWF) that the subsystem would have to wait before becoming urgent. Normally, .TAMIS should not be so small that most subsystems are penalized. It is intended only that the largest subsystems be so penalized. In fact, the site might want to set .TAMIS so high that no subsystem is penalized. If the site decides to set .TAMIS low enough to penalize a significant number of subsystem memory requests, care should be taken in setting .TALPP. (The Memory Wait Time Versus Requested Memory report of the TSS Response Time Analysis System will report the number of memory requests of various sizes and will make it possible to determine what percentages are .TAMIS and over. It will also provide

feedback on the time these subsystems must wait.)
.TALPP should not be set so large that many memory requests must wait longer than the acceptable response before they can become urgent. On the other hand, a .TALPP setting of "1" means no penalty at all.

Determine the current settings of these five parameters from system patches and Appendix I. Compare these values to the ranges suggested in the paragraphs above. If all values are in their range or are outside it in the desired direction of change (Table VII-2), an anomalous situation arises that cannot be resolved by these procedures. If one or more of the parameters is outside its range in a direction opposite that desired, change it along the lines recommended above. If it is impossible to change the parameter or the change does not increase TSS memory size, return to the beginning of Section VII.C. If a second execution of this section does not increase TSS memory size, proceed to Section VII.D.

i. Adjust .TASID. The existence of an urgent user will cause TSS to increase its memory size only if that user has been urgent at least .TASID seconds. In order to arrive at this procedure, the analyst has identified that urgent users exist but that few of them are urgent as long as .TASID. This may mean that .TASID is too long; it also may mean that users have to wait too long before they are classified as urgent.

Usually, .TASID should not be lowered to zero because of the way TSS treats urgent users. TSS attempts three actions on behalf of an urgent user: (1) reserve an area in TSS memory for the user (referred to as a "core fence"), (2) force swap any user in memory over A.MTQ seconds (default is seven seconds), and (3) increase TSS memory size. The first two actions are attempted immediately after the user becomes urgent. The last, more drastic action, is instituted only after .TASID seconds. The first two remedies are therefore given no chance to work if .TASID is lowered to zero.

The minimum value for .TASID depends on (1) the length of time it takes for the first two remedies to work, (2) the length of time the site defines as acceptable response, and (3) the relative abundance of memory. No minimum is specified in this document. The site will have to experiment with various values of .TASID.

Determine the site value of .TASID (Appendix I) from system patches and a listing of TSSA. Experiment with lower settings of .TASID. With settings under one second, the analyst should reduce A.SD3I to the same value as .TASID so that urgent users are recognized before .TASID is exceeded. This gives TSS a chance to force swap or to set up a core fence before increasing memory. Unnecessary memory increases are thereby avoided. If TSS memory size is not increased, retain a lower setting of .TASID and return to the beginning of Section VII.C. If a second execution of this section does not help, proceed to Section VII.D.

j. Adjust TSS Size Decrease Parameters. If the analyst reaches this section, excessive memory size decreases have been detected in Section VII.C.2 above. Selecting the default values of .TAMRI (time interval between possible memory reductions) and .TASRI (size of each reduction) would easily keep excessive decreases from happening; the solution is to set these parameters to the default values of five minutes and 5K words, respectively. See Appendix I for parameter locations and bit positions. If TSS size does not increase with these parameter changes, return to the beginning of Section VII.C or proceed to Section VII.D.

k. GCOS Denial of Additional Memory Requests. The TSS statistics that directed the analyst to this section indicated that the TSS memory wait time problem was probably due to denials by GCOS of TSS requests for memory. This may indicate a GCOS memory availability problem or a GCOS urgency code problem. However, the decision to come to this section was based on lack of evidence of internal TSS problems--not on any proof of actual GCOS denials of TSS memory requests. Use Patch Number 2 in Appendix I to count GCOS denials of TSS increase requests. If there are ever more than [100] over a day's period, assume GCOS denials of TSS memory requests are a problem, and proceed to paragraph (2) below. If these are never more than [100] counted, assume they are not a problem, and proceed to paragraph (1) below.

(1) Some TSS memory-related parameters are not likely to influence TSS memory size unless large changes are made in their values. For this reason, they are not treated in the other memory parameter

sections. These parameters should be examined in this section before the analyst requests professional CPE help. The parameters are listed in Table VII-3. The values of these parameters should be the default values or should differ from the default values only in the direction indicated in the "Desired Direction of Change" column of the table. See Appendix I for the patch locations and bit positions of these parameters. If any parameter differs from the default value in the direction opposite to the Desired Direction in Table VII-3, change its value back to the default value. If the subsystems at a site are unusually large (a significant percentage [30%] of entries in the Memory Sizes of Subsystems report over 20K), increasing .TAMII beyond the default may be advisable. Unless a dramatic improvement to TSS response takes place, proceed to Sections VII.D and VII.E.

TABLE VII-3. OTHER TSS MEMORY SIZE PARAMETERS

PARAMETER	DEFAULT VALUE	DESIRED DIRECTION OF CHANGE	DESCRIPTION
.TASCP	30 seconds	Lower	Minimum interval between urgent user size increases
.TAG	14 seconds	Lower	Minimum interval between any size increase
.TAMII	7K	Higher	Memory size increase amount

(2) The cause of GCOS denials of TSS memory requests is likely to be a special site operating constraint: one large, unswappable program in the same quadrant as TSS, special constraints on where TSS can be loaded, etc. Normal batch programs, even with very

high (i.e., 62) urgencies may be swapped to make room for TSS. Only those with the "dead bit" (bit 26 of the program state word, .STATE) set would normally be expected to keep TSS from expanding. Investigate any such causes that may exist at this site and attempt to change them to allow TSS to grow. If this is impossible, proceed to Sections VII.D and VII.E. If those sections do not help, and if more memory would allow TSS to grow when needed, consider increasing the total memory configured on the system.

D. ADJUST MEMORY PRIORITIES

When TSS response is unacceptable, it may be that unacceptable responses occur only when users execute subsystems requiring more memory than the average subsystem. Because TSS memory is normally allocated to the smaller subsystems first, and because larger subsystems must wait longer before becoming urgent, the normal problems of finding an available space in memory are compounded for these subsystems. Several parameters related to urgent users can be changed to give more priority to large subsystems. This section focuses on these parameters.

All subsystems are divided into two groups for memory allocation priorities. The parameter .TAMIS (default 36K) divides the two groups. The subsystems less than .TAMIS in size receive what this document calls "standard priority service." The subsystems of size .TAMIS or larger receive what this document calls "low priority service." The procedure below isolates subsystems with poor response and adjusts the memory allocation priorities to favor those subsystems. This adjustment may be to standard and/or low priority service, depending on whether the subsystems with poor response are under and/or over .TAMIS in size.

The remedies prescribed in these sections involve shifting priority from one subsystem size to another. Some previously acceptable responses may become unacceptable, especially if forced swapping and core fences are used too much. If this happens, it may mean that the problem is impossible to solve by rearranging priorities. However, it is possible to cut forced swapping by raising A.MTQ, and to control both forced swapping and core fences by raising .TASWT (making it harder for small subsystems to become urgent). If these remedies simply correct one response problem by creating another, proceed to Section VII.E.

1. Gather Data

Obtain the TSS Response by Subsystem report and the Memory Sizes of Subsystems report of the TSS Response Time Analysis System.

a. Date and Time

(1) Report Value. The Start Time, in the upper right corner of either report, gives the date and time of the data collection.

(2) Form Entry. Enter the date and time in the labeled boxes of the part of the form labeled "Subsystems Associated with Unacceptable Response." A separate copy of the form will be required for each separate date and time.

b. Subsystem Name

(1) Report Value. Using the TSS Response by Subsystem report, determine which subsystems logged unacceptable response according to the site definition of acceptable response. If the site definition of response time is not the same as the TSS Response Time Model definition (see Section II), the response times reported in the TSS Response by Subsystem report will be reported differently than they would under the site's definition of response time. If the main difference in definition is that the site considers response time to end at the first output to the user, compensation can be made for the difference in the following way. For each subsystem, note the Normal Output and Special Output time on that subsystem's Elapsed Time in Model States by Subsystem report. This time spent waiting for output to complete would not be part of the site-defined response time. Therefore, it may be subtracted from the responses recorded for that subsystem in the TSS Response by Subsystem report.

(2) Form Entry. Enter the names of the subsystems that logged unacceptable responses into the Name column.

c. Percentage Associated With the Subsystem.

(1) Report Value. The Percentage Associated With This Subsystem value reports the percentage of a subsystem's Total Response Time value (see the Total Response Time column) during which the subsystem was active. Because only one subsystem is active at a time for any one user, this subsystem was actually responsible for this percentage of the response time reported on its row.

(2) Form Entry. Enter the percentage in the Percentage column on the form.

d. Decision. Place a check next to each subsystem entered on the form that has a Percentage figure greater than [30%]. These subsystems are the ones mainly responsible for unacceptable response.

e. Mean Memory Size. Obtain the Memory Sizes of Subsystems report for each subsystem checked on the form.

(1) Calculation. Multiply the midpoint of each bucket (the midpoint of 7K-9K is 8K) times the value in the "PCT" column for that bucket. Sum those products and divide by 100. The result is the mean memory size for that subsystem. Many subsystems will have a constant size and the mean will simply be that size.

(2) Form Entry. Enter the mean size for each subsystem in the Mean Size column.

f. Overall Mean Size. Also obtain the Memory Sizes of Subsystems Report that is the summary for all subsystems.

(1) Calculation. Calculate the average size over all subsystems by using this report as if it were a report for a particular subsystem.

(2) Form Entry. Enter the mean size over all subsystems at the bottom of the Mean Size column on the form.

g. Maximum Standard Priority Size.

(1) Value. From the patches, a listing of TSSA, and Appendix I, determine the current value of .TAMIS.

(2) Form Entry. Enter the value on the form in the box provided in 1024-word blocks (example - 36K).

h. Decision. Repeat steps a. through g. for several data collection periods. If [any] mean size of a checked subsystem is smaller than, or the same as, the mean size for all subsystems, proceed to Section VII.E. Response time for these subsystems would be increased by the changes to memory priorities made in this section.

.TAMIS represents the memory size at which a subsystem is considered too large and is given a low priority for memory allocation. Two ranges of memory size are discussed here: (1) greater than the mean size over all subsystems, but less than .TAMIS and (2) greater than or equal to .TAMIS. If all individual subsystem means fall in the first range, follow the procedures in Section VII.D.2. If all the means fall in the second range, follow the procedures in Section VII.D.3. If the means fall in both ranges, follow the procedures in Section VII.D.2 and then follow the procedures in Section VII.D.3 if it is necessary to improve response further for those subsystems over .TAMIS in size.

2. Standard Priority Service

In order to be classified urgent, a user's subsystem must have waited longer than both of two time periods. The first is a constant--the value of .TASWT in seconds (default is three seconds). The second is the number of 512-word blocks the subsystem requires divided by .TASWF (default is eight). The dividend is expressed in seconds. Table VII-4 shows the waiting times at which a user is first classed urgent.

The purpose of this section is to give higher priority to subsystems at the upper memory sizes of standard priority service. This can be done by raising .TASWF so that a subsystem would have to be quite large to have to wait an extra second. It can also be done by raising .TASWT so that the small subsystems have to wait the same amount of time. It

is suggested that .TASWF first be tried at values of 16 and 24. This would lower the waiting times for a 32K subsystem from eight seconds to four seconds and three seconds, respectively.

TABLE VII-4. MEMORY WAIT PERIODS REQUIRED FOR URGENT USER STATUS

SUBSYSTEM SIZE (K WORDS)	WAIT PERIOD ¹ (SECONDS)
2K	3
8K	3
12K	3
16K	4
24K	6
32K	8

¹Assumes default parameter settings.

Experiment with selected values of .TASWT and .TASWF to find the best parameter settings. Obtain the TSS Response by Subsystem report for any periods of poor response to determine the effect of the change.

3. Low Priority Service

Any subsystem of size .TAMIS or larger will have to wait .TALPP times as long to become urgent. If standard priority service parameters and .TALPP are at their default values, a 36K subsystem would have to wait nine seconds to be urgent if .TAMIS were 40K; it would wait 36 seconds if .TAMIS were 36K.

Either .TAMIS or .TALPP can be changed to grant large subsystems higher priority. Changing .TAMIS to 200K or changing .TALPP to one (the default is four) would give all subsystems the standard priority service (standard, but not necessarily equal, depending on .TASWT and .TASWF). Setting .TALPP at two would give low priority service a little higher priority than a setting of four but would keep the priority lower than normal.

Deciding what settings to use for these parameters involves management objectives and the site's definition of what constitutes acceptable response. Various settings should be tried, since the results may not be predictable quantitatively. If acceptable response is not achieved, follow the procedures in Section VII.E below.

E. REDUCE MEMORY DEMAND

The objective of this section is to reduce the demand for TSS swap core. Since this reduction involves decisions and questions outside the scope of this document, only suggestions are made here.

This is the last section that concerns memory wait time. If none of these suggestions helps, try another substate of TSS response time (Section II).

1. List Largest Users

Use the Memory Sizes of Subsystems Report to determine the memory sizes of all subsystems. Multiply each size by the time spent at this size. This gives a figure that reflects both the size of memory used and how long it was used. A small part of memory used for a long time may lengthen memory wait times as much as a large part used for a short time. A list of the subsystems in their order of size-time products might suggest which of the actions below could best reduce memory demand. If a particular application or set of users appears responsible for most of the largest items on the list, the users involved may be able to fix the problem themselves or suggest ways to fix it.

2. Shift from One System to Another

Shifting work from one subsystem to a similar one (i.e., BASIC to FORTRAN) might reduce memory demand. One subsystem might use less memory and/or might use its memory for a shorter length of time. The Memory Sizes of Subsystems Reports for two subsystems doing the same work would indicate relative memory and time efficiencies.

3. Shift to Batch

Applications that involve little interaction and that are not time critical might be shifted to batch jobs.

4. Optimize User Code

Sites that use the FORTRAN (.YPO or FRTN) or BASIC (BASY) subsystems can try to influence users to write programs that use less memory or that use memory for shorter periods. An often-used subsystem could be recoded in GMAP and perhaps broken into several smaller subsystems grouped by the frequency of use of various parts of the code.

5. Reduce Disk I/O Time

Because subsystems are locked in swap core while they wait for disk I/O to complete, memory demand will be reduced if the time needed for I/O to disk is significantly reduced. Resolving disk I/O problems, and optimizing subsystems and user code so that both use fewer disk I/O's would help. This is true only if the time spent in the Disk I/O state of the TSS Response Time Model is significant.

6. Reduce CPU Time Required

The total demand for the CPU by TSS subsystems also influences how long these subsystems remain in swap core. Reducing the time subsystems and user code spend executing instructions will also reduce memory demand if significant amounts of time are spent in Eligible for CPU state of the TSS Response Time Model.

7. Reduce Non-Swappable Non-TSS Process Time

The common types of Non-TSS Process Time lock the user's subsystem in one place in core for the duration of the Non-TSS Process. Execute Section X if any significant time is spent in non-swappable Non-TSS Processes. Table X-1 lists which types of Non-TSS Processes are swappable.

SECTION VIII, GWAKE WAIT TIME SEARCH PROCEDURE

The procedures for analyzing GWAKE time should be used in nearly every response time tuning effort. The tuning steps proposed are inexpensive and effective, and relatively small amounts of GWAKE time can point to severe problems.

A. PROCEDURE SUMMARY

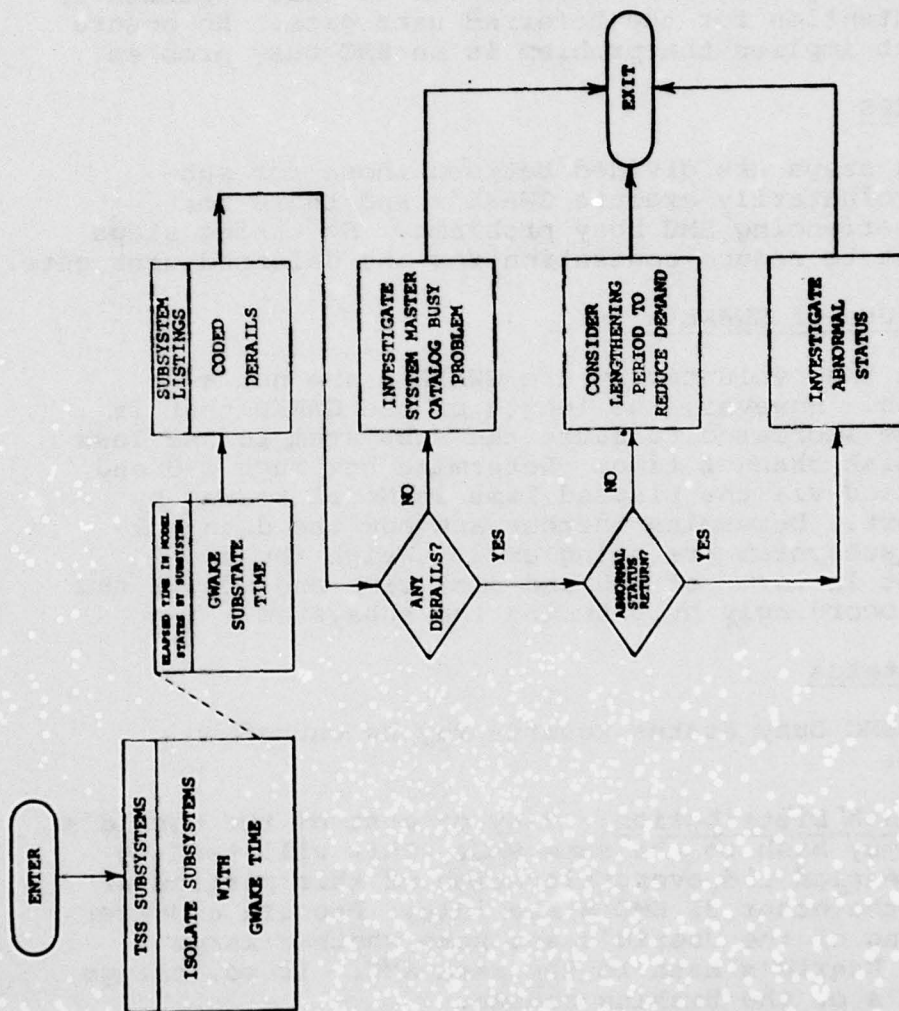
Figure VIII-1 charts the procedures described in this section. The report used here is the Elapsed Time Spent in Model States by Subsystem report.

Execution of a GWAKE Derail instruction causes a subsystem to be ignored for a period of time that is specified by the subsystem when the Derail is executed. A subsystem normally must request a GWAKE. It may be requested because the subsystem wishes to do nothing for a time. The TSS monitor subsystems TSAR and TSRI and the batch job monitor subsystem DJST use the GWAKE to sample data every few seconds. Subsystems may also use GWAKE's to output a message at regular intervals. Subsystems may use a GWAKE when a request to the executive is denied. The subsystem can try again later, when conditions may have changed.

The TSS executive can also cause a subsystem to use a GWAKE Derail. The TSS executive will make it seem as though the subsystem requested GWAKE time if: (1) the File System returns a System Master Catalog (SMC) busy status to a TSS request for File System action, or, (2) the user's subsystem tries to close the deferred user gate (.TSDGT) while the gate is already closed. (The SMC busy status informs TSS that some process [e.g., TSS user or batch job] is creating/purging/requesting access to/relinquishing access to some file or catalog emanating from the same SMC TSS wants to use. The SMC stays busy until that user finishes his change to the catalog structure. The SMC is not busy during a normal read/write to a file.)

B. ANALYZE GWAKE TIME

Use the Elapsed Time Spent in Model States by Subsystem report to isolate the subsystems that are responsible for the GWAKE time. DJST, TSAR, and TSRI can be expected to use GWAKE's, as can subsystems that do something periodically (e.g., write a message to the terminal). GWAKE time charged to these subsystems represents no problem, unless the subsystem consumes a significant amount of CPU time, channel



GWAKE WAIT TIME SEARCH PROCEDURE

FIGURE VIII-1

time, or memory space. Subsystems that have no reason to use GWAKE's may be experiencing error returns from the TSS executive functions or may be having problems with busy SMC's or deferred user gates. Examine a source listing of the subsystem and note any GWAKE Derails that are coded and why they are executed. If there are no Derails in the source, there must be a SMC busy problem or contention for the deferred user gate. Use Patch Number 11 (see Appendix I) to measure contention for the deferred user gate. No counts with this patch implies the problem is an SMC busy problem.

C. TUNING STEPS

The tuning steps are divided between those for subsystems that voluntarily execute GWAKE's and those for subsystems experiencing SMC busy problems. No tuning steps currently exist to reduce contention for the deferred user gate.

1. Voluntary Use of GWAKE's

Subsystems that voluntarily use GWAKE's are not a problem as such. However, the length of the GWAKE that is executed can be increased to cause the subsystem to use less CPU time and disk channel time. Determine how much CPU and disk time is used via the Elapsed Time in Model States by Subsystem report. Determine whether and how the data collected by the subsystem are being used. Weigh the use versus the cost in terms of CPU and disk time and adjust the GWAKE length accordingly by patching the subsystem.

2. SMC Busy Status

Excessive SMC Busy Status returns may be caused by several things.

a. Poor Hash Distribution. Many or most of the Userid's at a site may hash to the same SMC. This will tend to cause contention and overutilization of this particular SMC while the other 31 SMC's are idle. Procure a Master Save listing of the Userid's and note whether large numbers of Userid's hash to the same SMC. If so, change the Userid's or the hashing scheme.

b. File Restores. Any file restore (user or Master) may tend to lock one of the SMC's almost continuously. Any TSS users whose Userid's hash to the same SMC will experience degraded response time. Start a policy of not running Restores during periods of heavy TSS usage. Have operators TERM any file restores started by accident/subterfuge during those periods.

c. Poor Catalog Structure. Each catalog or subcatalog has space for recording links to 15 subcatalogs or files. More than this number of subcatalogs/files directly under a particular catalog causes one or more extensions to be allocated in which to store the extra links (19 subcatalogs or files per extension). Each disk access brings in one main record (15 subcatalogs/files) or one extension. This means that a set of 225 files cataloged directly under a Userid (i.e., "quick access" files) would take one main record and 12 extensions, with an average of six to seven disk accesses required to locate a file. Actually, the average might be higher since the newer and therefore more active files would be cataloged on the newer extensions, which will be further from the main catalog record. The same set of 225 files could be cataloged using 15 subcatalogs, each subcatalog containing 15 files. Each file locate would then require exactly two disk accesses; one to read the Userid record and get the subcatalog pointer, and the other to read the subcatalog record containing the file pointer. The smaller disk access requirement reduces response for the user locating the file, as well as reducing the time during which other users will find this particular SMC locked.

Educate users to look for this problem and check system catalogs and user catalogs to the extent possible and/or needed to eliminate the problem. One way this can occur is for extensive numbers of commands to be implemented using the command loader. Each command requires a separate quick access file under the same catalog (CMDLIB). Some of these commands could be implemented using command lists and primitives in TSSA, thus, eliminating/moving some of the files. Outmoded/never-used commands and/or files should be eliminated.

d. Excessive File Scrubbing. Excessive allocations/deallocations and excessive lengths of files that are allocated/deallocated often can cause unnecessary scrubbing (zeroing-out) of file space that is never used. The SMC involved in the allocation/deallocation stays locked during this procedure, which means that it may be locked five times longer than necessary if the space requested is five times larger than needed. As possible, reduce all files to the size actually used.

e. Important SSA Modules Not in Memory. A series of File Management System SSA modules takes control while an SMC is locked. The SMC will be locked longer if the heavily-used modules in this series are not in main memory. Follow the procedure in Section X.C to correct this problem. Do not restrict MCOUNT to TSS module usage only, because, in this case, the modules used by batch jobs are as important as the modules used by TSS.

SECTION IX. OUTPUT WAIT TIME SEARCH PROCEDURE

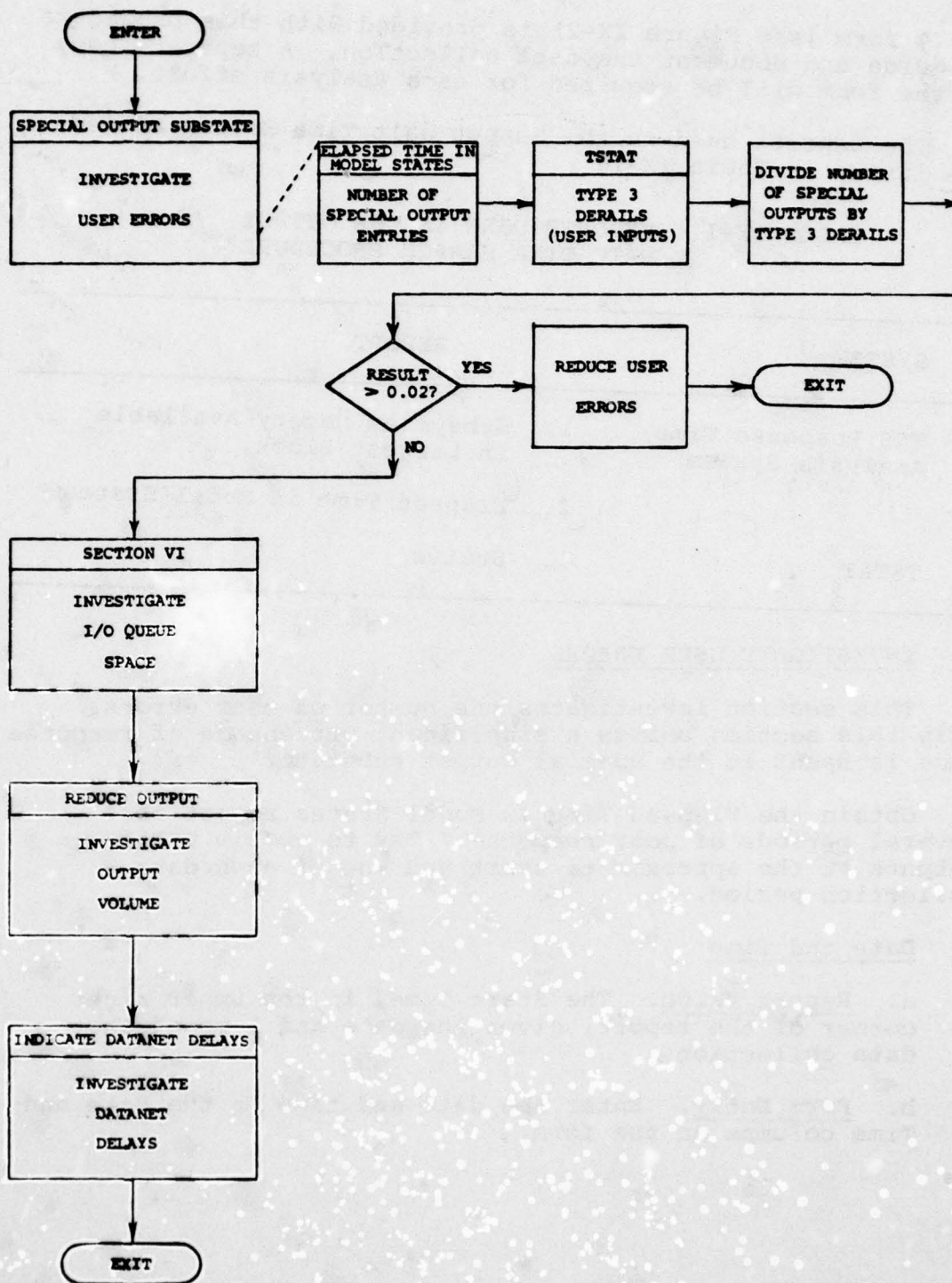
The procedures in this section analyze output wait time. These procedures should be used if response times are unacceptable and if the TSS Response Time Analysis system indicates that a major portion of response time is Normal Output Wait Time or Special Output Wait Time.

A. PROCEDURE SUMMARY

Output wait time is accumulated while a subsystem is blocked from execution waiting for output to the terminal to finish. Three basic conditions can block a subsystem from execution until output to the terminal is finished. One condition is a user error that results in a TSS executive error message. The user is in the Special Output substate while the error message is printed. Another condition involves the execution of certain Derails while output is in progress. Perhaps the most common occurrence of this is the Return Derail, signifying the subsystem is finished. The user waits in the Normal Output state until his output is finished. A third condition is that buffer space can be exhausted when more output is desired. Part of swap core is allocated for extra buffers (EBM) when the subsystem's original buffer has been filled. The extra buffer space holds almost 4,000 characters. Subsystems that fill or nearly fill this extra buffer space and then relinquish (via a Return Derail) may account for much of the output wait time. Sometimes the extra buffer space is not available. This is most likely to happen in a lightly loaded system with a very large subsystem that fills all available TSS memory.

Several conditions can decrease effective terminal speed and aggravate the previous three conditions. Insufficient I/O queue space can cause an output request to wait until queue space becomes available. Insufficient Datanet capacity may mean that output is delayed in the Datanet front-end processor (FEP). Inordinately long disk access times to the swap files may delay output because the extra buffers are stored on the swap files once the subsystem is swapped.

The Output Wait Time Search Procedure includes four steps: (1) Investigate User Errors, (2) Investigate I/O Queue Space, (3) Investigate Output Volume, and (4) Investigate Possible Datanet Delays. Figure IX-1 charts the steps for analyzing Output Wait Time.



OUTPUT WAIT TIME SEARCH PROCEDURE

FIGURE IX-1

A form (see Figure IX-2) is provided with this procedure to guide and document the data collection. A separate copy of the form will be required for each analysis effort.

The reports used in the Output Wait Time Search Procedure are listed in Table IX-1.

TABLE IX-1. REPORTS USED IN THE OUTPUT WAIT TIME SEARCH PROCEDURE

SYSTEM	REPORT
TSS Response Time Analysis System	1. Subsystem Memory Available in Largest Block 2. Elapsed Time in Model States
TSTAT	1. Status

B. INVESTIGATE USER ERRORS

This section investigates the number of user errors. Skip this section unless a significant percentage of response time is spent in the Special Output substate.

Obtain the Elapsed Time in Model States report for several periods of poor response. Try to obtain TSTAT outputs at the approximate start and end of each data collection period.

1. Date and Time

a. Report Value. The Start Time, in the upper right corner of the report, gives the date and time of the data collection.

b. Form Entry. Enter the date and time in the Date and Time columns on the form.

OUTPUT WAIT TIME SEARCH FORM

DATE:

USER ERRORS			
DATE	TIME	NUMBER OF ERRORS	USER INPUTS
MEANS			

OUTPUT WAIT TIME SEARCH FORM

FIGURE IX-2

2. Number of Errors

a. Report Value. Use the Number Entries value in the Special Output row.

b. Form Entry. Enter the value on the form in the Number of Errors column.

3. Number of User Inputs

a. Report Value. The number of Type 3 Derails (out of the 60+ types) executed gives an estimate of the number of separate user input messages.

b. Form Entry. Enter the value in the User Inputs column.

4. Calculation of Means

a. Calculations. Calculate the mean of the Number of Errors values and the mean of the User Inputs values over all data collection periods.

b. Form Entry. Enter the means at the bottoms of their respective columns.

5. Decision

If the mean Number of Errors value is less than [2%] of the mean User Inputs value, consider user errors to be low and proceed to Section IX.C. If not, attempt to reduce user errors. Interviews with users will provide clues about the kinds of errors users make. Information can then be distributed about avoiding such errors. Making modifications to the command language and its meaning may also help.

C. INVESTIGATE I/O QUEUE SPACE

These procedures and objectives coincide with those of Section VI.D. Execute that section and return here if I/O queue space is not a problem.

D. INVESTIGATE OUTPUT VOLUME

The objective of this procedure is to determine whether the volume of output to the terminals can be reduced.

AD-A073 133

COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON DC
WORLDWIDE MILITARY COMMAND AND CONTROL SYSTEM (WWMCCS). H-6000 --ETC(U)
SEP 78 B M WALLACK, G H GERO
CCTC-TM-180-78-VOL-3

F/G 9/2

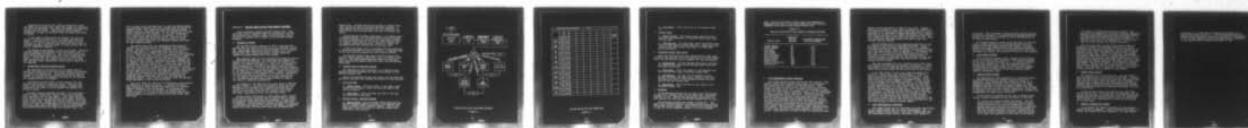
UNCLASSIFIED

NL

2 of 2

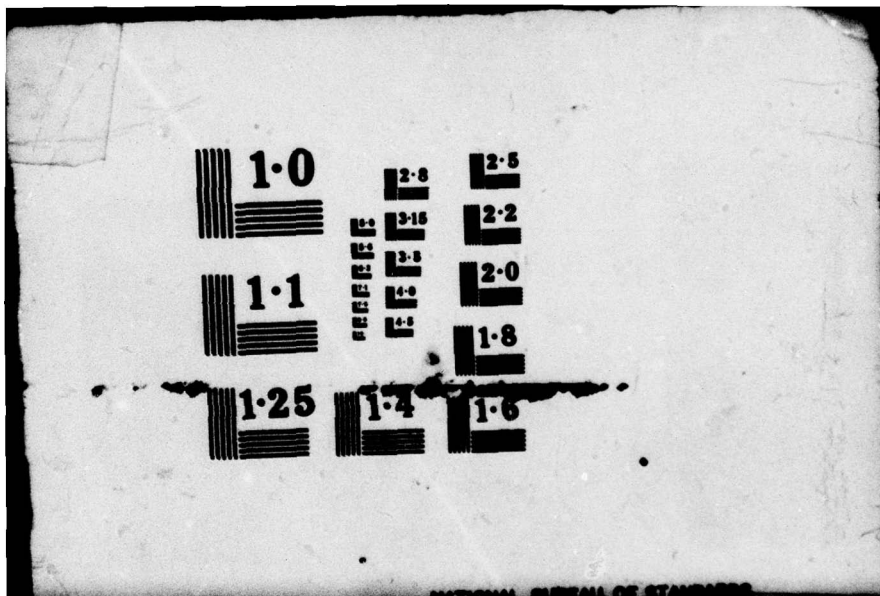
AD
A073 133

AD
A073 133



END
DATE
FILMED

9 79
DDC



Interviews with users will usually decide this issue. The TSTAT status report will report the number of characters of output by subsystem. This may help isolate the users who require the most output. There may be other commands, subsystems, or ways to structure files to reduce the volume of output.

If the output volume cannot be reduced, the site may weigh the costs and benefits of increasing the speed of output. One might try to reduce the time spent in other substates of the TSS Response Time Model and then return to consider upgrading terminal speeds if these efforts fail to achieve acceptable response.

The amount of improvement is hard to estimate. Doubling terminal speeds would probably halve Output Wait time; it could conceivably eliminate it. But the eliminated Output Wait time may become added user think time (Subsystem Idle state). If the user must read most or all of the output before entering the next command, faster output may mean that the user simply spends more time trying to catch up to the computer.

E. INVESTIGATE POSSIBLE DATANET DELAYS

The objective of this section is to suggest ways to explore the possibility that the Datanet front-end processor is a delay to terminal I/O. Since this document does not attempt to tune the Datanet, outside help must be used to isolate the problem and recommend solutions if the Datanet is a suspected source of delays.

Significant Datanet delays to TSS response are rare. They are also hard to isolate and prove. There are two "tests" that may help in deciding whether to commit more resources to an effort to prove whether the Datanet is a source of delay.

One test involves repeated entering of the same short (five or fewer characters) line of "code" with a numeric first character (for a line number). This should be done while the user is in the BIN (Build Input) mode, frequently called "star level" because TSS puts an asterisk (*) at the start of each line. Except for a buffer write to disk once every 6-20 input lines (depending on line and buffer length), TSS simply takes the input, adds it to an in-core buffer, sends another asterisk, and waits for further input. All

this is done in one courtesy call, so that the delays inside the mainframe should be negligible except when the buffer is dumped to disk. The number of lines input between buffer dumps should not change by more than one because the line and buffer lengths are not changing. If all long responses at this terminal are evenly spaced (for example, on the 7th, 13th, 19th, 26th, and 32nd lines input), they may be due to delays encountered trying to dump the buffer to disk. Any long response times that are unevenly spaced would tend to imply Datanet delays.

The second test calls for simultaneous monitoring of response by stopwatch and the TSS Response Time Analysis System. Because other users will probably be on the system at the same time, it is suggested that the "test user" use a subsystem (e.g., MASA) that other users are unlikely to use. This will make it possible to separate their responses from his. Have the test user go through a scenario likely to reproduce poor response of the type that is suspected to be due to Datanet delays. Carefully time each response with a stopwatch, using the TSS Response Time Model's definition of response time (see Section II). Either note when the TSS Response Time Analysis System begins to collect data, or use the Start Time option to restrict data reduction to include only the period during which responses were timed. Run a series of data reductions with only two responses on each reduction, so that each response may be individually compared to the stopwatch time. Compare the stopwatch timings with the responses recorded for that user in the TSS Response by Subsystem Report. Significant differences (over two seconds) would presumably be due to Datanet delays.

Note that these two tests do not necessarily prove whether the Datanet is a source of delays. If delays are noted, they conceivably could emanate from some error in the GCOS modules handling I/O to the Datanet. If no delays are noted, it could be that the Datanet was temporarily not causing delays, or it could be that the Datanet delays only certain types of interactions and the test user never used that type.

SECTION X, NON-TSS SERVICE WAIT TIME SEARCH PROCEDURE

These procedures analyze Non-TSS Process time. They should be used if response times are unacceptable and the TSS Response Time Analysis System indicates that a major portion of response time is spent in the Non-TSS Process substate.

A. PROCEDURE SUMMARY

The Non-TSS Process substate is used to describe subsystems that are waiting for service to be performed outside TSS. The normal types of service performed are File Management System (FMS) functions (create, delete, grow, gain access to files) and interrogations of GCOS functions (USERID from LOGN, SNUML entry from SYSOUT).

The response times of the interrogations depend primarily on conditions outside TSS. The time they take depends on general CPU time availability, I/O device and channel availability, and memory availability. Tests for the absence of these resources are provided in the Batch Turnaround Time Analysis Procedures. CPU time availability can be influenced by the type of priority the job in question is given (Priority B, I/O Priority, or Urgency Throughput). Memory availability can be influenced by the urgency of the program. Procedures for analyzing the interrogations' wait time are based on (1) the urgency and priority of the program involved and (2) the Batch Turnaround Time Analysis tests for memory, CPU, and I/O resources availability.

Processing for the FMS functions takes place in FMS SSA modules (.MFSxx). The speed of processing these functions depends on the locations of the modules involved and the availability of the SSA within TSS as well as the availability of processor time and disk channel time to TSS. Busy SMC's (see Section VIII) can also delay them.

Unlike delays encountered in other model states, delays to Non-TSS Process functions tend to cause delays to other users besides those executing the functions. A user cannot be swapped or moved during most Non-TSS Process functions. Consequently, certain blocks of memory lie idle for relatively long periods and fragment the rest of TSS memory, increasing memory wait times for other TSS users and increasing TSS

memory size. The FMS functions lock SMC's, causing other users and batch jobs to encounter SMC busy statuses. A large amount of Non-TSS Process time can, therefore, be more serious than the same amount of time in another sub-state.

This procedure includes seven parts: (1) Isolate Type of Non-TSS Process, (2) File Management System Processes, (3) GCOS Interrogation Processes, (4) Line Length Process, (5) Special Batch Process, (6) Normal Batch Process, and (7) Console Interaction Process. Figure X-1 charts the procedure parts that analyze Non-TSS Process time.

A form (see Figure X-2) is provided with this procedure to guide and document the data collection. A separate copy of the form will be required for each analysis effort.

The Non-TSS Process Time report and the Elapsed Time Spent in Model States by Subsystem report are used in this section. The Batch Turnaround Time Analysis Procedures tests used are the Memory Constraint Test, and the Urgency Codes Test.

B. ISOLATE TYPE OF NON-TSS PROCESS

The objective of this procedure is to identify which types of Non-TSS Processes are major contributors to the Non-TSS Process time.

Obtain the Non-TSS Process Time report for the same time periods as the data that brought the analyst to this section.

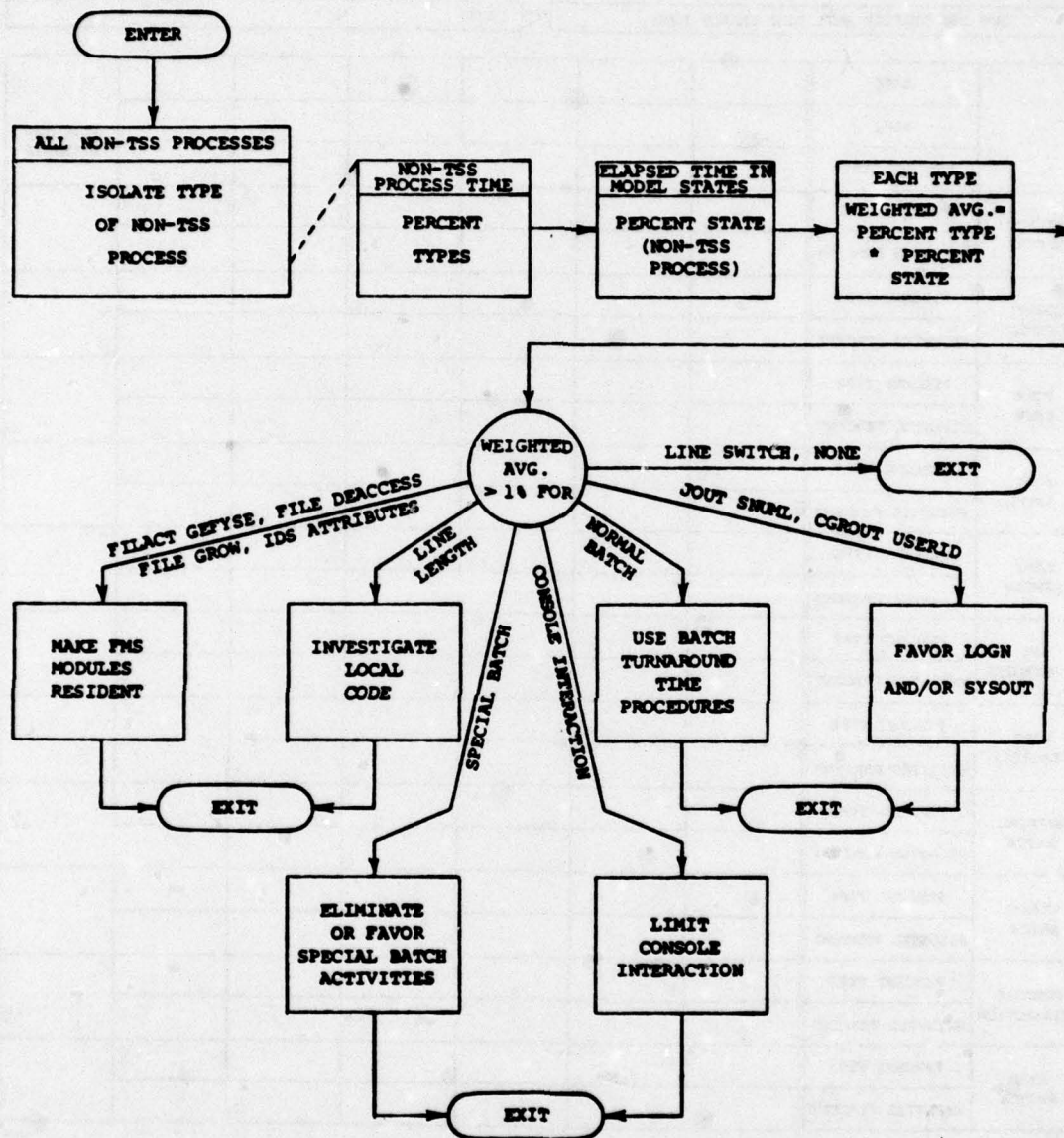
1. Date and Time

a. Report Value. The Start Time, in the upper right corner of the report, gives the date and time of the data collection.

b. Form Entry. Enter the date and time in the Date and Time rows on the form.

2. Percent State: Non-TSS Process

a. Report Value. Use the Percent State value for the Non-TSS Process substate recorded on the Elapsed Time spent in Model States (summary) report. This is the same report used in Section II when the analyst was directed to this section.



NON-TSS SERVICE WAIT TIME SEARCH PROCEDURE

FIGURE X-1

NON-TSS SERVICE WAIT TIME SEARCH FORM						DATE:	
	DATE						WEIGHTED AVERAGES
	TIME						
	PERCENT STATE						
FILACT GEFYS	PERCENT TYPE						
	WEIGHTED PERCENT						
CGROUT USERID	PERCENT TYPE						
	WEIGHTED PERCENT						
FILE GROW	PERCENT TYPE						
	WEIGHTED PERCENT						
JOUT SMUGL	PERCENT TYPE						
	WEIGHTED PERCENT						
LINE LENGTH	PERCENT TYPE						
	WEIGHTED PERCENT						
IDS ATTRIBUTES	PERCENT TYPE						
	WEIGHTED PERCENT						
FILE DEACCESS	PERCENT TYPE						
	WEIGHTED PERCENT						
SPECIAL BATCH	PERCENT TYPE						
	WEIGHTED PERCENT						
NORMAL BATCH	PERCENT TYPE						
	WEIGHTED PERCENT						
CONSOLE INTERACTION	PERCENT TYPE						
	WEIGHTED PERCENT						
LINE SWITCH	PERCENT TYPE						
	WEIGHTED PERCENT						

NON-TSS SERVICE WAIT TIME SEARCH FORM

FIGURE X-2

b. Form Entry. Enter the value in the Percent State row.

3. Percent Type

a. Report Values. The Percent Type values for each type give the percentage of total Non-TSS Process time of that type.

b. Form Entries. For each type, enter the Percent Type value in that type's Percent Type row. Repeat the Date, Percent State and Percent Type steps for each time period monitored.

4. Weighted Percent Type Means

This step calculates the mean percentage for each type, weighting the data for each time period by the Percent State value for the Non-TSS Process substate for the time period.

a. Calculation. For each Percent Type value, multiply the Percent Type value by the Percent State value for the same time period.

b. Form Entry. Enter the result in the Weighted Percent row for that type of Non-TSS Process Time.

c. Calculation. For each row of Weighted Percent Values, calculate the mean of the values in that row. The mean will serve as the Weighted Average for that type of Non-TSS Process Time.

d. Form Entry. Enter the mean at the right of its Weighted Percent row.

5. Decision

If the Weighted Average for any type of Non-TSS Process exceeds [1%], consider that type of wait high. Otherwise, consider it low. Execute the appropriate subsections of this section for each type of Non-TSS Process that is high.

The associated subsections appear in Table X-1. The table also indicates whether the subsystem awaiting the Non-TSS Process can be swapped. Since Line Switch Process time is part of no TSS response problem, it is not discussed

here. During Line Switch Process time, the subsystem is swapped and the user is entering commands to and receiving responses from some on-line system outside TSS.

TABLE X-1.
NON-TSS SERVICE WAIT TIME SEARCH PROCEDURE SECTIONS

TYPE OF WAIT	SUBSYSTEM CAN BE SWAPPED	ASSOCIATED PROCEDURES SECTION NUMBERS
FILACT GEFYSE	NO	X.C
CGROUT USERID	NO	X.D
File Grow	NO	X.C
JOUT SNUML	NO	X.D
Line Length	NO	X.E
IDS Attributes	NO	X.C
File Deaccess	NO	X.C
Special Batch	YES	X.F
Normal Batch	YES	X.G
Console Interaction	YES	X.H
Line Switch	YES	None

C. FILE MANAGEMENT SYSTEM PROCESSES

The FILACT GEFYSE process occurs when TSS executes a MME GEFYSE instruction to satisfy the request of a subsystem that executed a DRL (derail) FILACT instruction. This process is used to access files, modify catalog structure, create and purge files, etc. The File Grow process is used to add more disk space to permanent files. The IDS Attributes process is used to obtain necessary information for accessing an Integrated Data Store (IDS) file. The File Deaccess process is used to deaccess files (mark them idle). All four of these processes are accomplished by executions of various File Management System (FMS) SSA modules. Several FMS modules may be used each time one of these processes is involved, and the same module may be executed more than once, necessitating its being copied to disk and then back into memory. The execution, read from disk, execution, copy to disk, read from disk, execution sequence is strictly serial - each execution or disk I/O must be completely ended

before the next execution or disk I/O can occur. Besides reducing the number of times these processes are needed, the main method for eliminating some of the time spent in these processes is to make some of these FMS modules "core resident." Then they do not have to be loaded from disk each time they are invoked and they do not have to be copied to disk. Sites with significant TSS usage should have one or more core resident FMS SSA modules.

Use the MCOUNT option or the Mass Store Monitor to count the number of executions of each FMS SSA module. Those executed the most are prime candidates for core residence. An analyst may wish to limit the counts to executions by TSS only (use patch number three in Appendix I). An Analyst may also want to trace certain very important paths (such as File Grow, if the Non-TSS Process Time report shows inordinate lengths of time spent in that process) through the code to determine exactly what FMS modules will be used to accomplish that process.

Once the candidates for core residence are picked, have them loaded into hard core at Startup, if possible. Include the modules in the \$ LOAD section of the startup deck, or patch the Startup program and the entry word of each module.

If no room exists in hard core, load the modules into TSS core. Apply patch four (see Appendix I) to TSS to enable the private SSA module code in TSS0 to work. Then apply patch six to cause TSS to load the modules. (While this can be done using commercial releases of GCOS, it may not be possible with all WWMCCS releases.) If the modules cannot be loaded into TSS, increase the size of SSA cache. A final possibility is to alter the modules into TSS (say, between TSSN and TSS0) and relink TSS. Note that the entry words of the modules must be modified to show the existence and location of these private SSA modules. Note also that the addresses and address fields of any patches to TSS0 must be changed to reflect the new location of TSS0 and any patches to the FMS modules must become patches to TSS as well or be altered into the TSS version of the modules. The TSS modules will be used by TSS; all other programs will use the disk-resident versions of these modules.

D. GCOS INTERROGATION PROCESSES

The CGROUT USERID and the JOUT SNUML processes interrogate parts of GCOS outside TSS for needed information. The CGROUT USERID process obtains a new user's USERID from LOGN. The JOUT SNUML process obtains information from SYSOUT on

job output. Both processes accomplish their aims by putting an entry in the queue of the proper GCOS system program and waiting for a return to be made to TSS by courtesy call or an entry in TSS's queue.

No accepted method to speed these processes is currently known; further understanding and experience with actual problems involving these processes are needed. One suggestion is to make SYSOUT and/or LOGN permanently resident (do not allow them to swap) if the problem is of sufficient importance to justify the allocation of core for this purpose.

E. LINE LENGTH PROCESS

The Line Length process simply changes the line length field in a table in GCOS hard core. This process requires no I/O and no passive resource (i.e., it cannot be delayed by file busy conditions). The one possible delay is a closed gate, which cannot delay the process significantly. If this process consumes enough time to become a problem, research the local versions of the code executed and/or request professional help.

F. SPECIAL BATCH PROCESS

The objective of this procedure is to reduce the amount of time spent waiting for special batch activities to terminate. The simplest and most effective remedy is to reduce the number of special batch activities. If users recompile the same interactive programs frequently, they should be shown how to save the compiled form of the program so that it can be run again without recompiling. It may also be possible to switch to a different compiler or language that accomplishes the compilation within TSS.

If the above two remedies do not work, a special batch activity could be speeded through the system in two ways:

- (1) Raise its relative urgency for memory. The subsystem sets the urgency of the special batch activity. Currently TSS will not allow the subsystem to submit a special batch activity with an urgency over 40 (decimal). Raising the urgency higher than 40 requires patches to both TSS and the subsystems involved. Another way to raise the relative urgency of special batch activities is to lower the urgencies of batch jobs in the system.

(2) Raise its relative priority for the CPU. The priority of a special batch activity could be changed by switching the dispatching algorithm to Urgency Throughput (presumably its urgency is relatively high) or perhaps to I/O Priority, depending on the amount of I/O done by other jobs in the workload mix. This can be done while TSS retains its Priority B status.

Like all batch jobs, Special Batch Process jobs are slowed by the type of bottlenecks treated by the Batch Turnaround Time Analysis Procedures. Executing several of the tests from those procedures may help cut Special Batch Non-TSS Process time. The Memory Constraint Test (which references three other tests), is especially recommended. The Urgency Codes Test would be changed somewhat to test for too many jobs with urgency codes approximately equal to or higher than the Special Batch Process jobs. Note that these jobs bypass the System Scheduler, the Peripheral Allocator, and SYSOUT. They may not appear on the Batch Turnaround Time Analysis System reports, or may appear in an erroneous manner. This may make it impossible to execute the Turnaround Time Model Scan procedure on their behalf.

G. NORMAL BATCH PROCESS

The submission of a batch job (i.e., through CARDIN) does not usually entail any Non-TSS Process time. A Normal Batch Non-TSS Process occurs only when the user informs TSS that he wants to wait until the batch job is finished. A response time problem caused by long Normal Batch Non-TSS Process times is probably a batch turnaround time problem. Use the Batch Turnaround Time Analysis Procedures in Volume II of the Guide.

It may be that users do not realize the reason they are waiting so long. They should be informed that they are running and waiting for a batch job, not a TSS interaction. Perhaps they do not actually intend to wait for the batch job to finish, but intend merely to start the job and go on to other work.

H. CONSOLE INTERACTION PROCESS

It is unusual for Console Interaction to contribute significantly to response. Console interactions should, of course, be allowed only when necessary. For that reason,

the Derail to initiate,them is normally privileged (i.e., only a Master user can do it). The site should examine the console log and determine whether the interaction is necessary. If it is, some change to the system might make it unnecessary. It can then be made impossible by patches to TSSK.

DISTRIBUTION

Addressee

Copies

CCTC Code

C124 (Reference and Record Set)	3
C124 (Stock)	6
C126 (Reference and Record Set)	2
C702	350

DCA Code 205	1
------------------------	---

Defense Documentation Center, Cameron Station, Cameron Station, Alexandria Virginia 22314 . .	2
--	---

364

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TM 180-78	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) H-6000 Tuning Guide WWMCCS SYSTEM TUNING PROCESS Volume III		5. TYPE OF REPORT & PERIOD COVERED
7. AUTHOR(s) Barry M. Wallack George H. Gero JR.		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Command and Control Technical Center CPE, (C702), The Pentagon, Rm BE685 Washington, D.C. 20301		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Command and Control Technical Center CPE, (C702), The Pentagon, R, BE685 Washington, D.C. 20301		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE 1 September 1978
		13. NUMBER OF PAGES 109
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
<div style="border: 1px solid black; padding: 5px; text-align: center;"> DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited </div>		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Tuning, H-6000, WWMCCS, Computer, Performance Evaluation		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Federal Computer Performance Evaluation and Simulation Center (FEDSIM) has developed a document for WWMCCS installations that can be used by site personnel to analyze the performance characteristics of their Honeywell 6000 (H-6000) computer systems. This document, called an <u>H-6000 Tuning Guide</u> , incorporates detailed analysis procedures that guide the analyst in applying specific techniques to improve system performance. (cont'd on reverse)		

DD FORM 1473 1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE 98

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The four volumes of the H-6000 Tuning Guide present a precisely structured system of procedures for the analysis of the performance of WWMCCS computer services and systems:

- Volume I WWMCCS System Tuning Process. The first volume describes the overall structure and application of the Tuning Guide. It explains the approach, procedures, and processes taken by the Tuning Guide to provide analyses of batch job turnaround time and GCOS Time Sharing System (TSS) response time.
- Volume II Batch Turnaround Time Analysis Procedures. The second volume presents a set of procedures for analysis of batch job turnaround time. It first presents a model of the processes and queue points associated with batch job turnaround time and then describes nine tests that use the model to direct the analysis of turnaround time.
- Volume III TSS Response Time Analysis Procedures. The third volume serves the same general purpose and has the same general structure as volume II. Volume III presents a complete set of procedures for investigating the response time of GCOS Time Sharing System (TSS) interactions. The volume first presents a model of the processes and queue points associated with TSS response time and then describes eight tests to direct an analysis of TSS response time.
- Volume IV H-6000 Tuning Guide Appendices. The fourth volume provides the appendices referenced by the other volumes of the Tuning Guide. The volume contains detailed descriptions of report formats and other referenced data.